ALIBABA CLOUD

阿里云

配置审计

自定义规则

文档版本：20210324

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.使用函数计算新建规则

当配置审计提供的托管规则不能满足您的需求时，您可以使用函数计算新建自定义规则。当您的自定义规则被触发时，配置审计会运行对应的规则函数，并给出规则合规结果。

前提条件

请确保您已开通函数计算服务，操作方法请参见开通函数计算服务。

背景信息

自定义规则与预设规则差异的差异如下：

- 托管规则

  托管规则是配置审计已在函数计算中构建的规则函数，使用时您可以直接在配置审计控制台选择所需函数。配置审计支持的托管规则，请参见托管规则列表。

- 自定义规则

  自定义规则需要您提前在函数计算中定义规则函数，使用时在配置审计控制台选择规则函数的ARN。

新建自定义规则

1. 在函数计算控制台新建函数。

   新建函数的操作方法，请参见新建函数。

   > **说明**
   > - 新建函数时，**创建方式**选择**事件函数**，**运行环境**选择python3，**函数入口**使用默认值index.handler，入口函数为handler。
   > - 如果您想深入了解函数计算，请参见函数计算。

2. 在函数计算控制台查看新建函数ARN。

   新建函数后，自动生成一个函数ARN，您可以在该函数的**概览**页面查看。

   

3. 在配置审计控制台新建规则时，选择新建函数ARN。新建规则的操作方法，请参见通过函数计算新建规则。

规则函数的代码构建

规则的本质是一段逻辑判断代码，这段代码存放在新建的函数中。在实际持续审计中，通过触发该函数的执行来实现资源评估。本函数代码主要有两个函数，具体如下：

- handler

  handler为入口函数，即自定义规则触发时调用的函数。handler在新建函数时进行设置。

- put_evaluations

  put_evaluations在handler中调用，返回合规结果。JSON样例如下：

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
import logging
import json
from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.auth.credentials import StsTokenCredential
from aliyunsdkcore.acs_exception.exceptions import ClientException
from aliyunsdkcore.acs_exception.exceptions import ServerException
from aliyunsdkcore.request import CommonRequest
logger = logging.getLogger()
# 用于筛选资源类型，以英文逗号（,）分隔多个资源类型。
MATCH_RESOURCE_TYPES = ''
# 合规类型
COMPLIACE_TYPE_COMPLIANT = 'COMPLIANT'
COMPLIACE_TYPE_NON_COMPLIANT = 'NON_COMPLIANT'
COMPLIACE_TYPE_NOT_APPLICABLE = 'NOT_APPLICABLE'
COMPLIACE_TYPE_INSUFFICIENT_DATA = 'INSUFFICIENT_DATA'
def handler(event, context):
  """
  处理函数
  :param event: 事件
  :param context: 上下文
  :return: 评估结果
  """
  # 校验Event
  evt = validate_event(event)
  if not evt:
    return None
  rule_parameters = evt.get('ruleParameters')
  result_token = evt.get('resultToken')
  invoking_event = evt.get('invokingEvent')
  # 初始化返回值
  compliance_type = COMPLIACE_TYPE_NOT_APPLICABLE
  annotation = None
  # 获取ConfigurationItem
  configuration_item = invoking_event.get('configurationItem')
  if not configuration_item:
    logger.error('Configuration item is empty.')
    return None
  ordering_timestamp = configuration_item.get('captureTime')
  resource_id = configuration_item.get('resourceId')
  resource_type = configuration_item.get('resourceType')
  # 获取评估结果
  compliance_type, annotation = evaluate_configuration_item(
    rule_parameters, configuration_item)
```

```
    # 评估结果
    evaluations = [
      {
        'complianceResourceId': resource_id,
        'complianceResourceType': resource_type,
        'orderingTimestamp': ordering_timestamp,
        'complianceType': compliance_type,
        'annotation': annotation
      }
    ]
    # 回写评估结果
    put_evaluations(context, result_token, evaluations)
    return evaluations
def evaluate_configuration_item(rule_parameters, configuration_item):
    """
    评估逻辑
    :param rule_parameters: 规则参数
    :param configuration_item: 配置项
    :return: 评估类型，注解
    """
    # 初始化返回值
    compliance_type = COMPLIACE_TYPE_NOT_APPLICABLE
    annotation = None
    # 获取资源类型和资源ID
    resource_type = configuration_item['resourceType']
    full_configuration = configuration_item['configuration']
    # 判断资源类型
    if MATCH_RESOURCE_TYPES and resource_type not in MATCH_RESOURCE_TYPES.split(','):
      annotation = 'Resource type is {}, not in {}.'.format(
        resource_type, MATCH_RESOURCE_TYPES)
      return compliance_type, annotation
    # 判断配置信息
    if not full_configuration:
      annotation = 'Configuration is empty.'
      return compliance_type, annotation
    # 转换为JSON
    configuration = parse_json(full_configuration)
    if not configuration:
      annotation = 'Configuration:{} in invald.'.format(full_configuration)
      return compliance_type, annotation
    # ========== 用户代码 start ========== #
    # ========== 用户代码 end ========== #
    return compliance_type, annotation
def validate_event(event):
    """
    校验Event
    :param event: Event
    :return: JSON对象
    """
    if not event:
      logger.error('Event is empty.')
    evt = parse_json(event)
    logger.info('Loading event: %s .' % evt)
    if 'resultToken' not in evt:
```

```
        logger.error('ResultToken is empty.')
        return None
    if 'ruleParameters' not in evt:
        logger.error('RuleParameters is empty.')
        return None
    if 'invokingEvent' not in evt:
        logger.error('InvokingEvent is empty.')
        return None
    return evt
def parse_json(content):
    """
    JSON类型转换
    :param content: JSON字符串
    :return: JSON对象
    """
    try:
        return json.loads(content)
    except Exception as e:
        logger.error('Parse content:{} to json error:{}.'.format(content, e))
        return None
def put_evaluations(context, result_token, evaluations):
    """
    回调配置审计OpenAPI回写评估结果
    :param context: 函数计算上下文
    :param result_token: 回调令牌
    :param evaluations: 评估结果
    :return: None
    """
    # 需要输入当前阿里云账号的AccessKey ID和AccessKey Secret，需要具备AliyunConfigFullAccess权限。
    client = AcsClient(
        'XXXX',
        'XXXXXXX',
        'cn-shanghai',
    )
    # 新建request，并设置参数，domain为config.cn-shanghai.aliyuncs.com。
    request = CommonRequest()
    request.set_domain('config.cn-shanghai.aliyuncs.com')
    request.set_version('2019-01-08')
    request.set_action_name('PutEvaluations')
    request.add_body_params('ResultToken', result_token)
    request.add_body_params('Evaluations', evaluations)
    request.set_method('POST')
    try:
        response = client.do_action_with_exception(request)
        logger.info('PutEvaluations with request: {}, response: {}.'.format(request, response))
    except Exception as e:
        logger.error('PutEvaluations error: %s' % e)
```

规则函数的入参

在函数中设置的入参信息保存在ruleParameters中，其他内容在规则触发时自动生成事件信息。JSON格式如下：

```
{
  version:"版本号",
  orderingTimestamp:"命令执行开始时间",
  invokingEvent:{
   messageType:"消息类型",
    configurationItem：{
     "accountId":"阿里云账号ID",
      "arn":"资源ARN",
      "availabilityZone":"可用区",
      "regionId":"地域ID",
      "configuration":"字符串形式的资源本身配置信息，各类资源有所不同",
      "configurationDiff":"配置变更内容",
      "relationship":"关系",
      "relationshipDiff":"关系内容变更",
      "captureTime":"捕获时间",
      "resourceCreationTime":"资源创建时间",
      "resourceStatus":"资源状态",
      "resourceId":"资源ID",
      "resourceName":"资源名称",
      "resourceType":"资源类型",
      "supplementaryConfiguration":"补充配置",
      "tags":"标签"
    },
    notificationCreationTimestamp:"事件消息生成时间"
  },
  ruleParameters：{
   "key":"value"
  },
  resultToken:"用户在函数计算中的回调信息"
}
```

context参数是上下文信息，规则触发时自动带入。

- `context.credentials.access_key_id:"accessKey"`
- `context.credentials.access_key_secret:"accessSecret"`
- `context.region:"地域信息"`