Config

Best Practices

Issue 01

Date 2025-01-22

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: https://www.huaweicloud.com/intl/en-us/

Contents

1 Creating Rules

Overview

This example shows how to use the Java SDK to create and query Config rules.

When you create a **rule**, you need to specify a resource scope that the rule applies to.

Prerequisites

1. You have obtained the Huawei Cloud SDK and installed the Java SDK.
2. You have a Huawei Cloud account and an access key ID (AK) and a secret access key. You can view or create an AK/SK pair in **My Credentials** > **Access Keys** on the Huawei Cloud console. For details, see **Access Keys**.
3. **Config SDK** supports Java JDK 1.8 or later.

Installing the SDK

You can obtain and install the SDK using Maven. To use Maven, add dependencies to the **pom.xml** file. For details about SDK versions, see **SDK Center**.

```
<dependency>
    <groupId>com.huaweicloud.sdk</groupId>
    <artifactId>huaweicloud-sdk-config</artifactId>
    <version>{sdk-version}</version>
</dependency>
```

Example Code

```
public class CreatePolicyAssignmentDemo {
    public static void main(String[] args) {
        // There will be security risks if the AK and SK used for authentication is written into code. Encrypt the AK/SK and store them into the configuration file or environment variables.
        // In this example, the AK and SK are stored in environment variables. Before running this example, set environment variables HUAWEICLOUD_SDK_AK and HUAWEICLOUD_SDK_SK.
        String ak = System.getenv("HUAWEICLOUD_SDK_AK");
        String sk = System.getenv("HUAWEICLOUD_SDK_SK");
        String regionId = "<region id>";
        HttpConfig config = HttpConfig.getDefaultHttpConfig();
        config.withIgnoreSSLVerification(true);

        ICredential auth = new GlobalCredentials().withAk(ak).withSk(sk);
        ConfigClient client = ConfigClient.newBuilder().withHttpConfig(config).withCredential(auth)
            .withRegion(ConfigRegion.valueOf(regionId)).build();
```



```
        CreatePolicyAssignmentsRequest createRequest = new CreatePolicyAssignmentsRequest()
            .withBody(new
PolicyAssignmentRequestBody().withPolicyAssignmentType(PolicyAssignmentRequestBody.PolicyAssignment
TypeEnum.BUILTIN)
                .withName("<Your policy_assignment_name>").withDescription("<Your
policy_assignment_description>")
                .withPolicyFilter(new PolicyFilterDefinition()).withPolicyDefinitionId("<Your
policy_definition_id>"));

        try {
            CreatePolicyAssignmentsResponse createResponse = client.createPolicyAssignments(createRequest);
            System.out.println(createResponse.toString());
            ShowPolicyAssignmentRequest showPolicyAssignmentRequest = new
ShowPolicyAssignmentRequest()
                .withPolicyAssignmentId(createResponse.getId());
            ShowPolicyAssignmentResponse showResponse =
client.showPolicyAssignment(showPolicyAssignmentRequest);
            System.out.println(showResponse.toString());
        } catch (ConnectionException | RequestTimeoutException | ServiceResponseException ex) {
            System.out.println(ex);
        }
    }
}
```

Response

```
class CreatePolicyAssignmentsResponse {
    policyAssignmentType: "policyAssignmentType"
    id: "id"
    name: "name"
    description: "description"
    policyFilter: class PolicyFilterDefinition {}
    period: "period"
    state: "state"
    created: "created"
    updated: "updated"
    policyDefinitionId: "policyDefinitionId"
    customPolicy: "customPolicy"
    parameters: {}
    createdBy: "createdBy"
}

class ShowPolicyAssignmentResponse {
    policyAssignmentType: "policyAssignmentType"
    id: "id"
    name: "name"
    description: "description"
    policyFilter: class PolicyFilterDefinition {}
    period: "period"
    state: "state"
    created: "created"
    updated: "updated"
    policyDefinitionId: "policyDefinitionId"
    customPolicy: "customPolicy"
    parameters: {}
    createdBy: "createdBy"
}
```

Reference

For more details, see **Resource Compliance Overview**.

Change History

Release On	Issue	Description
2024-12-25	1.0	This is the first official release.

2 Querying Resource Details, Relationships, and Change Records

Overview

This example shows how to use the Java SDK to query resource details, relationships, and change records.

1. **Resource List** only displays some resource attributes. The following example shows how to **query more details about a resource**.
2. **Associated Resources** displays **relationships between your Huawei Cloud resources**.
3. **Resource Timeline** records **resource changes**. A record will be added to the resource timeline when a service reports a resource attribute or relationship change to Config. Config retains resource change records for seven years by default.

Prerequisites

1. You have obtained the Huawei Cloud SDK and installed the Java SDK.
2. You have a Huawei Cloud account and an access key ID (AK) and a secret access key. You can view or create an AK/SK pair in **My Credentials** > **Access Keys** on the Huawei Cloud console. For details, see **Access Keys**.
3. **Config SDK** supports Java JDK 1.8 or later.

Installing the SDK

You can obtain and install the SDK using Maven. To use Maven, add dependencies to the **pom.xml** file. For details about SDK versions, see **SDK Center**.

```
<dependency>
    <groupId>com.huaweicloud.sdk</groupId>
    <artifactId>huaweicloud-sdk-config</artifactId>
    <version>{sdk-version}</version>
</dependency>
```

Example Code

```
public class ShowResourceRelationDemo {
    public static void main(String[] args) {
```



```
        // There will be security risks if the AK and SK used for authentication is written into code. Encrypt the AK/SK and store them into the configuration file or environment variables.
        // In this example, the AK and SK are stored in environment variables. Before running this example, set environment variables HUAWEICLOUD_SDK_AK and HUAWEICLOUD_SDK_SK.
        String ak = System.getenv("HUAWEICLOUD_SDK_AK");
        String sk = System.getenv("HUAWEICLOUD_SDK_SK");
        String regionId = "<region id>";
        HttpConfig config = HttpConfig.getDefaultHttpConfig();
        config.withIgnoreSSLVerification(true);

        ICredential auth = new GlobalCredentials().withAk(ak).withSk(sk);
        ConfigClient client = ConfigClient.newBuilder().withHttpConfig(config).withCredential(auth)
            .withRegion(ConfigRegion.valueOf(regionId)).build();

        try {
            String resourceId = "<resource id>";
            // Querying resource details
            ShowResourceDetailRequest resourceDetailRequest = new ShowResourceDetailRequest()
                .withResourceId(resourceId);
            System.out.println(client.showResourceDetail(resourceDetailRequest));
            // Querying resource relationships
            ShowResourceRelationsRequest resourceRelationsRequest = new ShowResourceRelationsRequest()
                .withResourceId(resourceId)
                .withDirection(ShowResourceRelationsRequest.DirectionEnum.IN);
            System.out.println(client.showResourceRelations(resourceRelationsRequest).toString());
            // Querying resource change records
            ShowResourceHistoryRequest resourceHistoryRequest = new ShowResourceHistoryRequest()
                .withResourceId(resourceId);
            System.out.println(client.showResourceHistory(resourceHistoryRequest).toString());
        } catch (ConnectionException | RequestTimeoutException | ServiceResponseException ex) {
            System.out.println(ex);
        }
    }
}
```

Response

```
class ShowResourceDetailResponse {
    id: 81fi****a864
    name: zh****ng
    provider: iam
    type: users
    regionId: global
    projectId:
    projectName:
    epId: 0
    epName: default
    checksum: 522u****e689
    created: 2023-09-18T12:56:30.000Z
    updated: 2023-09-18T12:56:30.000Z
    provisioningState: Succeeded
    state: Normal
    tags: {}
    properties: {pwd_status=false, pwd_strength=high, group_list=[f588****54c5], role_list=[], last_login_time=2023-09-18T12:57:45Z, virtual_mfa_device=false, login_protect={enabled=false}, credentials=[], policy_list=[], access_mode=default, is_root_user=false, enabled=true}
}

class ShowResourceRelationsResponse {
    relations: [class ResourceRelation {
        relationType: contains
        fromResourceType: iam.groups
        toResourceType: iam.users
        fromResourceId: f587****54c5
        toResourceId: 81fa****a864
    }]
    pageInfo: class PageInfo {
        currentCount: 1
        nextMarker: null
```



```
    }
}

class ShowResourceHistoryResponse {
    items: [class HistoryItem {
        domainId: 39f4****ea39
        resourceId: 81fa****a864
        resourceType: iam.users
        captureTime: 2023-09-21T15:39:27.632Z
        status: ResourceChanged.CREATE
        relations: [class ResourceRelation {
            relationType: isContainedIn
            fromResourceType: iam.users
            toResourceType: iam.groups
            fromResourceId: 81fa****a864
            toResourceId: b04e****8dd2
        }]
        resource: class ResourceEntity {
            id: 81fa****a864
            name: zh****ng
            provider: iam
            type: users
            regionId: global
            projectId:
            projectName:
            epId: 0
            epName: default
            checksum: 00ce****f053
            created: 2023-09-18T12:56:30Z
            updated: 2023-09-18T12:56:30Z
            provisioningState: Succeeded
            state: null
            tags: {}
            properties: {pwd_status=false, pwd_strength=high, group_list=[b04e****8dd2], role_list=[],
virtual_mfa_device=false, login_protect={enabled=false}, credentials=[], policy_list=[], access_mode=default,
enabled=true}
        }
    }]
    pageInfo: class PageInfo {
        currentCount: 1
        nextMarker: null
    }
}
```

Reference

For more details, see **Viewing Resource Changes**.

Change History

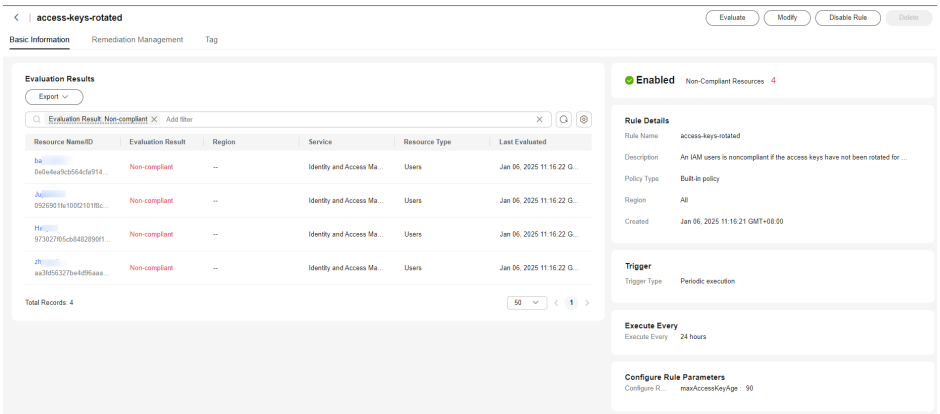
Release On	Issue	Description
2024-12-25	1.0	This is the first official release.

3 Creating Alarm Rules for Noncompliant Resources with Cloud Eye

Cloud Eye enables you to receive alarms when there are noncompliant resources detected by Config. You can query alarms on the Cloud Eye console. You can also configure an SMN topic to enable notification with Cloud Eye.

Applicable Scenario

This example uses the **access-keys-rotated** rule to see if all IAM users in an account have their access keys rotated within a specified time. Some IAM users may be detected noncompliant as shown in the following picture.



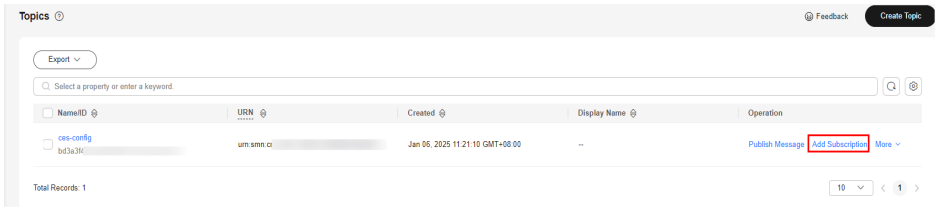
Step 1: Create a Rule.

1. Log in to the **Config Console**.
2. In the navigation pane on the left, choose **Resource Compliance**.
3. On the **Rules** tab, click **Add Rule**.
4. On the **Basic Configurations** page, select **access-keys-rotated** and click **Next**.
5. On the **Configure Rule Parameters** page, use the default value for **Execute Every**, select **All** for **Resource Scope**, and click **Next**.
6. Confirm the configurations and click **Submit**.

On the **Rules** tab, you can view the evaluation result of the created rule.

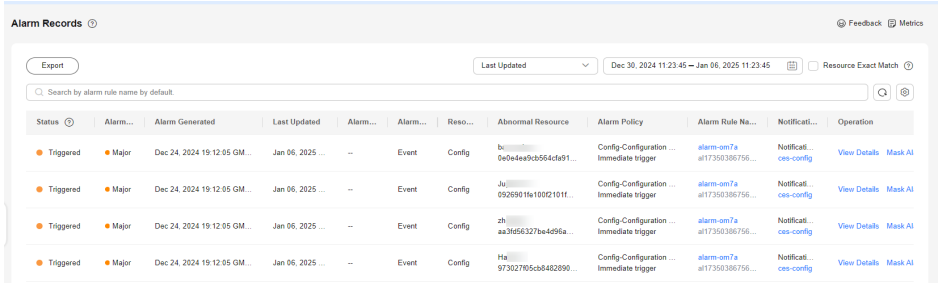
Step 2: Configure an SMN Topic

1. Log in to the **SMN console**.
2. In the navigation pane on the left, choose **Topic Management** > **Topics**.
   a. In the upper right corner, click **Create Topic**.
   b. Configure the **Topic Name** and **Display Name**, and click **OK**.
3. Add a subscription to the topic.
   a. On the **Topics** page, click **Add Subscription** in the **Operation** column for the created topic.
   b. Select **SMS** for the **Protocol**.
   c. Enter one or multiple mobile numbers.
   d. Click **OK**.
4. In the navigation pane on the left, select **Topic Management** > **Subscriptions** and click **Request Confirmation**.
5. Confirm subscription with the added mobile number.



Step 3: Set Alarm Notifications

1. Log in to the **Cloud Eye console** and set the region to **AP-Singapore**.
2. In the navigation pane on the left, select **Alarm Management** > **Alarm Rules**.
3. On the **Alarm Rules** page, click **Create Alarm Rule**.
   a. Configure the **Name**.
   b. Select **Event** for **Alarm Type**.
   c. Select **System event** for **Event Type**.
   d. Select **Config** for **Event Source**.
   e. Select **Configure manually** for **Method**.
   f. Enable **Alarm Notification** and select the SMN topic created in step 2 for **Notification Object**. Remain default settings for other parameters.
   g. Select **Generated alarm** for **Trigger Condition**.
   h. Click **Create**.
4. Check SMN messages or alarms on the Cloud Eye console to see if there are noncompliance resources detected by Config rules you created.



Related Links

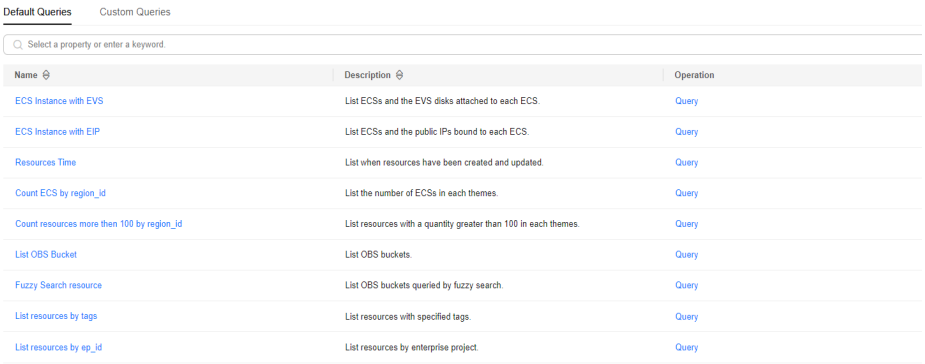
- **Viewing Events**
- **Creating an Alarm Rule to Monitor an Event**

4 Using Advanced Queries

In this section, you will learn how to use advanced queries to query resources and download resource data.

Applicable Scenario

You can use ResourceQL to query resources with the advanced query function provided by Config. Advanced queries make it convenient to export resource data as needed.

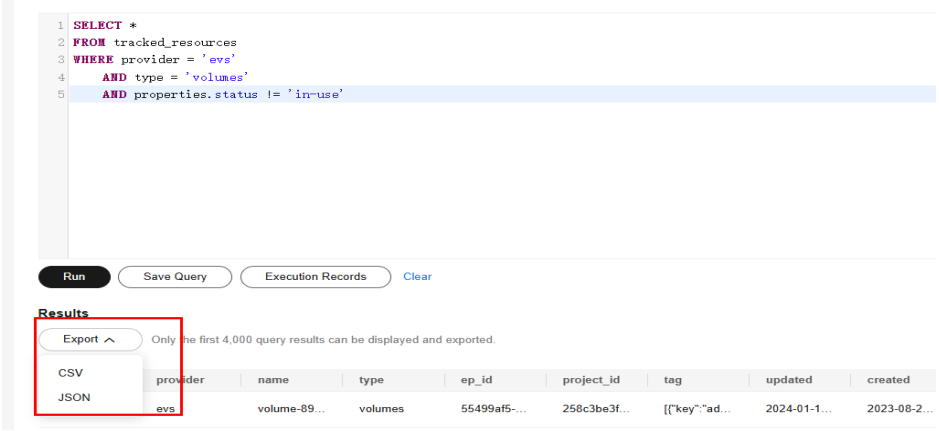


Procedure

1. Log in to the **Config Console**.
2. In the navigation pane on the left, select **Advanced Queries**.
3. Click **Custom Queries** and click **Create Query** in the upper right corner.
4. Enter the following query and click **Run** to query idle EVS disks.

```
SELECT *
FROM tracked_resources
WHERE provider = 'evs'
    AND type = 'volumes'
    AND properties.status != 'in-use'
```

5. On the **Results** area, click **Export** to export query results to a CSV or a JSON file.



Related Links

- ResourceQL Syntax Overview
- Querying Schemas

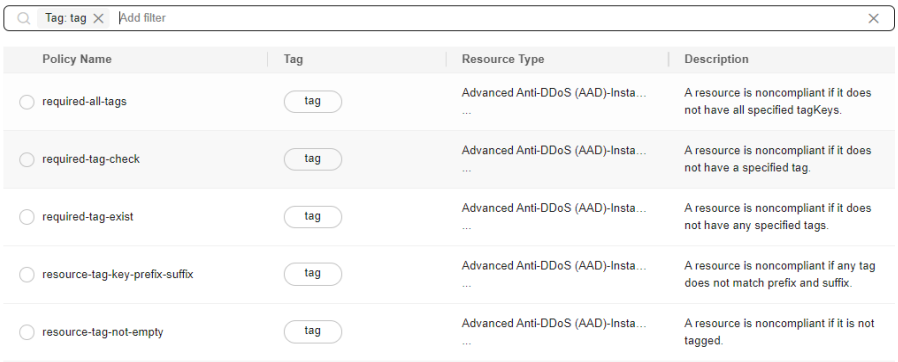
5 Querying Resources That Do Not Have Specific Tags

This section describes how to query resources that are not attached with certain tags.

Applicable Scenario

After a company moves to the cloud, as cloud resources keep growing, they usually need to manage hundreds of thousands or millions of resources within one account. You can use Tag Management Service (TMS) to classify and group resources by department, region, or project.

Config helps you identify resources that are not correctly tagged.

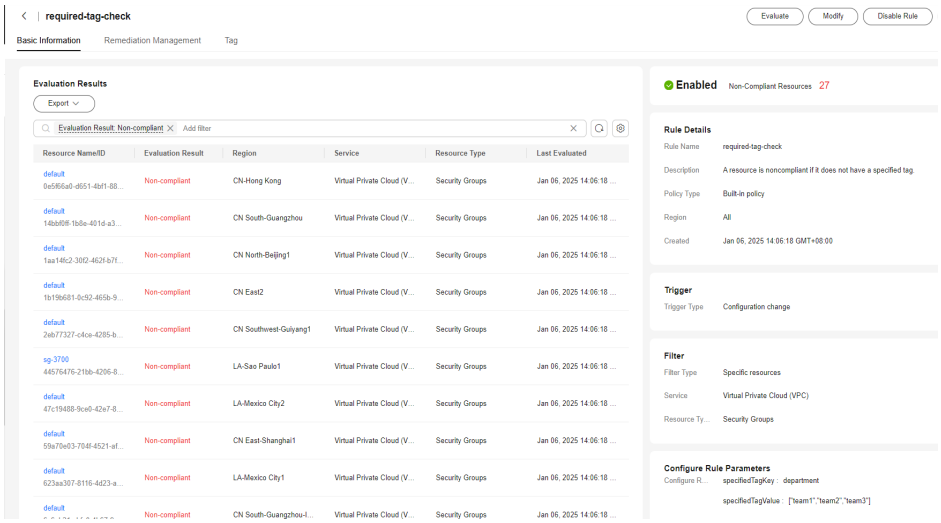


Procedure

1. Log in to the **Config Console**.
2. In the navigation pane on the left, choose **Resource Compliance**.
3. On the **Rules** tab, click **Add Rule**.
4. On the **Basic Configurations** page, select **required-tag-check** in the **Built-in Policy** area, and click **Next**.
5. On the **Configure Rule Parameters** page, select **Virtual Private Cloud (VPC)** for **Service**, **Security Groups** for **Resource type**, and **All** for **Region**.
6. Set **specifiedTagKey** to **department** and **specifiedTagValue** to **["team1", "team2", "team3"]**.

7. Confirm the configurations and click **Submit**.

On the **Rules** tab, you can view the evaluation result of the rule.



Built-in policies

Policy	Description
required-all-tags	If a resource does not have all the specified tags attached, this resource is noncompliant.
required-tag-exist	If a resource is missing any of the specified tags, this resource is noncompliant.
resource-tag-key-prefix-suffix	If a resource does not have any tags that are specified with specific key prefixes and suffixes, this resource is not compliant.
resource-tag-not-empty	If a resource is not tagged, this resource is noncompliant.
required-tag-check	If a resource does not have the specified tag attached, this resource is noncompliant.

6 Ensuring Resource Compliance by Tag, Region, and Organization

When implementing cloud resource compliance, enterprises usually face the following problems:

- The security requirements of the production environment are different from those of the test environment.
- Different regions have different requirements of resource compliance due to varying laws and regulations.

In addition, all accounts in an organization are required to be configured with unified security baseline requirements. Config provides a wealth of built-in policies to help enterprises manage resource compliance based on different scenarios.

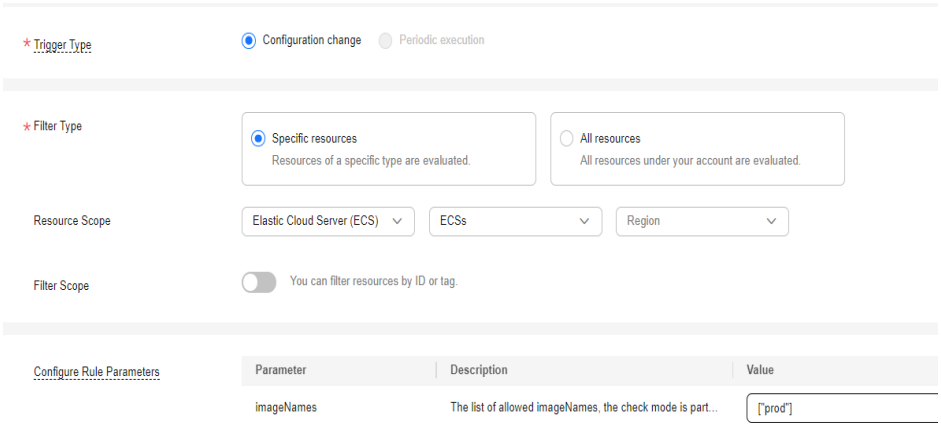
Evaluating Resources By Tag

Prerequisite: Your resources have been tagged. For details, see **Principles for Naming Tags**.

Scenario: If you tag resources by environment, you can use these tags to audit resources in different environments. Assume that you have added the tag key: **Env:Prod** to all resources in the production environment and **Env:Test** to all resources in the test environment. You can define tag values as needed.

Procedure

1. Log in to the **Config Console**.
2. In the navigation pane on the left, choose **Resource Compliance**.
3. On the **Rules** tab, click **Add Rule**.
4. Select a built-in policy, for example, **allowed-images-by-name**, and click **Next**.
5. On the **Configure Rule Parameters** page, remain the default settings for **Resource Scope** and select **All** for **Region**.
6. Toggle on **Filter Scope**, click **Tag**, and enter **Env** for **Tag key** and **Prod** for **Tag value**.



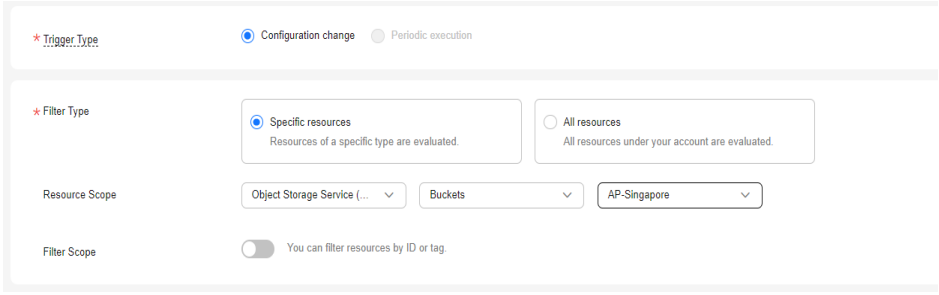
7. Click **Submit**. The rule is intended to evaluate the specified resource type in the production environment.
8. Return to the **Rules** tab to view evaluation results.

Evaluating Resources By Region

Scenario: If you do not want your OBS buckets in the regions outside of the Chinese mainland to be publicly accessed, you can create a rule to check if your OBS buckets are correctly configured to meet your exceptions.

Procedure

1. Log in to the **Config Console**.
2. In the navigation pane on the left, choose **Resource Compliance**.
3. On the **Rules** tab, click **Add Rule**.
4. On the **Basic Configurations** page, select **obs-bucket-public-read-policy-check** in the **Built-in Policy** area, and click **Next**.
5. On the **Configure Rule Parameters** page, remain the default settings for **Resource Scope** and select **AP-Singapore** for **Region**.



6. Click **Submit**. The rule is intended to evaluate your OBS buckets in the **AP-Singapore** region.
7. Return to the **Rules** tab to view evaluation results.

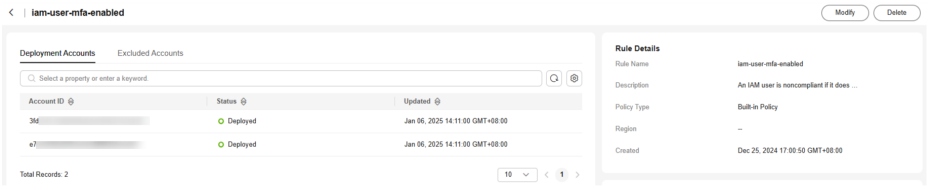
Creating an Organization Rule

Prerequisites: You have created an organization and you are the organization administrator or a delegated administrator of Config. For more details, see **Overview of Organizations** and **Specifying, Viewing, or Removing a Delegated Administrator**.

Scenario: You can deploy an **organization rule** to some or all member accounts in your organization. Generally, the security administrator that is not in charge of service specific tasks is responsible for deploying organization rules.

Procedure

1. Log in to the **Config Console**.
2. In the navigation pane on the left, choose **Resource Compliance**.
3. On the **Organization Rules** tab, click **Add Rule**.
4. Select a built-in policy, for example, **iam-user-mfa-enabled**, and click **Next**.
5. On the **Configure Rule Parameters** page, remain the default settings for **Resource Scope** and click **Next**.
6. Return to the **Organization Rules** tab to view evaluation results.



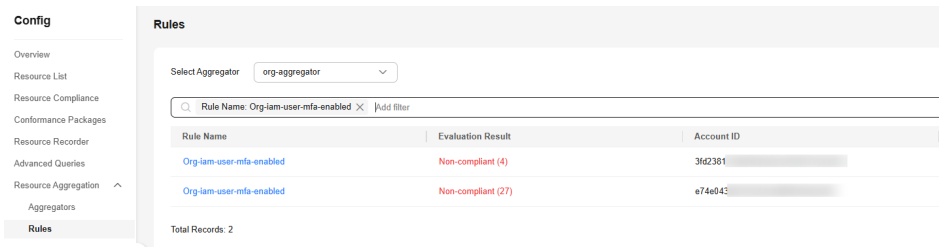
Aggregating Resource Data from an Organization

Prerequisites: You have created an organization and you are the organization administrator or a delegated administrator of Config. For more details, see **Overview of Organizations** and **Specifying, Viewing, or Removing a Delegated Administrator**.

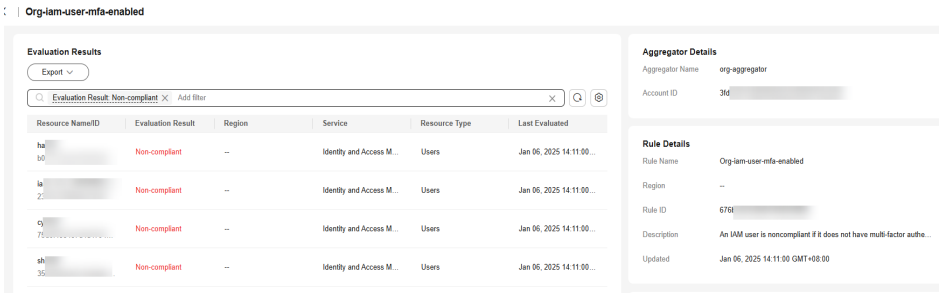
Scenario: The security administrator of an organization can create **aggregators** to query rules deployed to and resource compliance of member accounts in the organization.

Procedure

1. Log in to the **Config Console**.
2. In the navigation pane on the left, choose **Resource Aggregation** > **Aggregators**.
3. On the **Aggregators** page, click **Create Aggregator**.
4. Select **Allow data replication**, enter an aggregator name, set the **Source Type** to **Add my organization**, and click **OK**.
5. On the **Rules** page, select the created aggregator to view rules aggregated from the member accounts.



6. Click a rule name to view its evaluation results.



FAQs

Why Is There No Organization Rule Page on Config Console?

The organization rule function is only available to an organization administrator or an organization member who is a delegated administrator of Config.

Why Is an Organization Rule Abnormal After Being Deployed?

The resource recorder has not been enabled in the member account.

To get full functionality of Config, you need to enable the resource recorder. If the resource recorder is disabled, you may have problems using rules and conformance packages.

To deploy organization rules or conformance packages to member accounts, the resource recorder must be enabled for both the organization administrator or the delegated administrator of Config and all the involved members.

7 Automating Resource Management

In this section, you will learn how to automate the detection and remediation of noncompliant resources with Config rules and the remediation function. This ensures that noncompliant resources whether resulting from intended or unintended actions can be remediated within minutes, enhancing resource security.

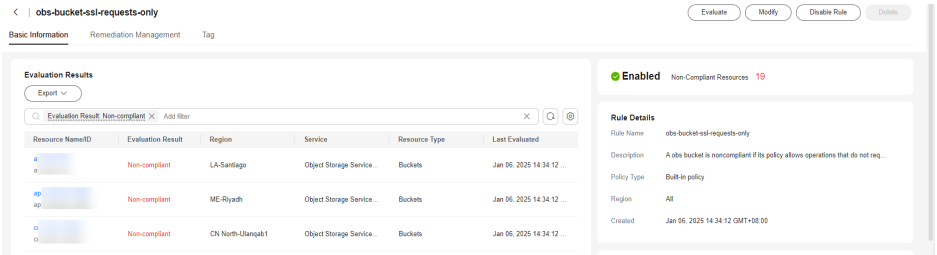
Applicable Scenario

Scenario: You can configure OBS bucket policies to block HTTP access to your buckets. For details **Keeping Data in Transit Safe**

Procedure

Create a rule.

1. Log in to the **Config Console**.
2. In the navigation pane on the left, choose **Resource Compliance**.
3. On the **Rules** tab, click **Add Rule**.
4. On the **Basic Configurations** page, select **obs-bucket-ssl-requests-only** in the **Built-in Policy** area, and click **Next**.
5. On the **Configure Rule Parameters** page, remain the default settings for **Resource Scope** and select **All** for **Region**. Click **Next** to confirm the configurations and click **Submit**
6. Return to the **Rules** tab to view evaluation results.



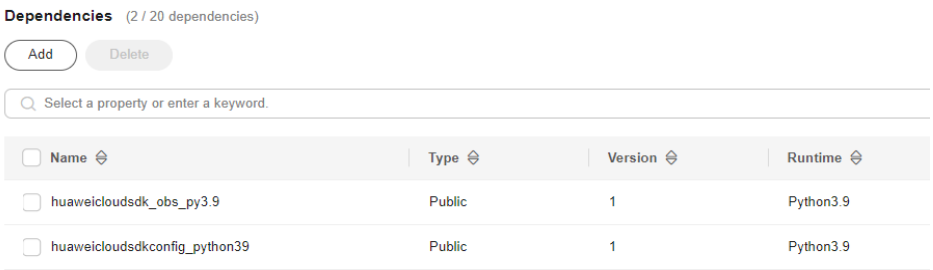
Configure mediation.

The following procedure shows how to use a FunctionGraph function to configure remediation. Python is used.

1. Log in to the **FunctionGraph console**.
2. In the navigation pane on the left, click **Functions** > **Function List**.
3. On the **Functions** tab, click **Create Function**.
4. Select **Event Function** for **Function Type**, select an agency, and select **Python 3.9** for **Runtime**. The agency selected must contain at least the following permissions:

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "obs:bucket:PutBucketPolicy",
        "obs:bucket:GetBucketPolicy",
        "rms:resources:get"
      ]
    }
  ]
}
```

5. After the function is created, add the two dependencies: **huaweicloudsdk_obs** and **huaweicloudsdkconfig**, to the function.



6. Add the following code to index.py:

```
import json

from obs.client import ObsClient
from huaweicloudsdkcore.auth.credentials import GlobalCredentials
from huaweicloudsdkconfig.v1.region.config_region import ConfigRegion
from huaweicloudsdkconfig.v1.config_client import ConfigClient
from huaweicloudsdkconfig.v1 import ShowResourceDetailRequest

def get_resource_region(context, domain_id, resource_id):
    auth = GlobalCredentials(
        ak=context.getSecurityAccessKey(),
        sk=context.getSecuritySecretKey(),
        domain_id=domain_id
    ).with_security_token(context.getSecurityToken())
    client = ConfigClient.new_builder() \
        .with_credentials(credentials=auth) \
        .with_region(region=ConfigRegion.value_of(region_id="cn-north-4")) \
        .build()
    resource = client.show_resource_detail(ShowResourceDetailRequest(resource_id)).to_json_object()
    return resource.get("region_id")

def getBucketPolicy(obsClient, bucket_name):
    resp = obsClient.getBucketPolicy(bucket_name)
    if resp.status < 300:
        print("Get Bucket Policy Succeeded")
        return resp.body.policyJSON
    if resp.status == 404 and resp.errorCode == "NoSuchBucketPolicy":
        print("NoSuchBucketPolicy")
        return "{\"Statement\": []}"
    assert False, f"Get Bucket Policy Failed: {resp.errorCode} | {resp.errorMessage}"
```

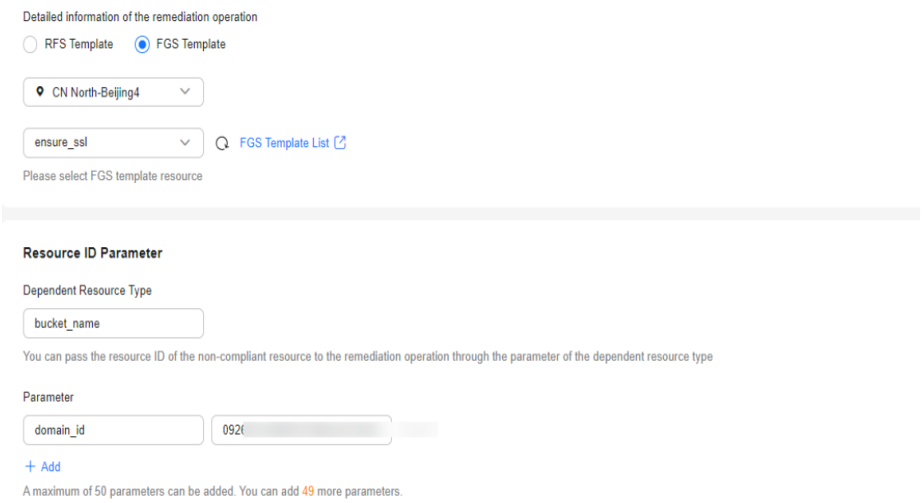


```
def ensurePolicySSL(obsClient, bucket_name, policy):
    policy["Statement"] = policy["Statement"] + [{
        "Sid": "ensure_secure_transport",
        "Effect": "Deny",
        "Principal": {"ID": ["*"]},
        "Action": ["*"],
        "Resource": [bucket_name, bucket_name + "/*"],
        "Condition": {"Bool": {"g:SecureTransport": ["false"]}}
    }]
    resp = obsClient.setBucketPolicy(bucket_name, policy)
    if resp.status < 300:
        print("Set Bucket Policy Succeeded")
    else:
        print(policy)
        assert False, f"Set Bucket Policy Failed: {resp.errorCode} | {resp.errorMessage}"

def handler(event, context):
    domain_id = event.get("domain_id")
    bucket_name = event.get("bucket_name")
    print("domain_id", domain_id)
    print("bucket_name", bucket_name)

    region_id = get_resource_region(context, domain_id, bucket_name)
    print("region_id", region_id)

    server = f"https://obs.{region_id}.myhuaweicloud.com"
    obsClient = ObsClient(
        access_key_id=context.getSecurityAccessKey(),
        secret_access_key=context.getSecuritySecretKey(),
        server=server,
        security_token=context.getSecurityToken()
    )
    policy = getBucketPolicy(obsClient, bucket_name)
    policy = json.loads(policy)
    ensurePolicySSL(obsClient, bucket_name, policy)
    obsClient.close()
```

7. (Optional) Modify basic settings: **Memory (MB)** and **Execution Timeout (s)**, and configure **Log** for the function. You are recommended to complete this step to ensure smooth resource remediation and enable logging in case of any errors that may occur.

Configuring Remediation

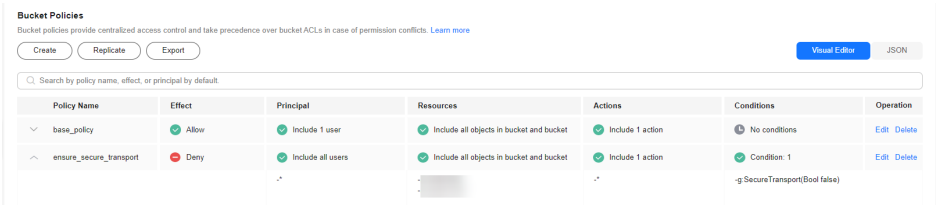
1. Log in to the **Config Console**.
2. In the navigation pane on the left, choose **Resource Compliance**.
3. On the **Rules** tab, click the name of the rule.
4. Click **Remediation Management** and click **Remediation Configuration**.
5. Select **Automatic** or **Manual** for **Method** and remain the default settings for **Retry Time Limit** and **Retries**.
6. Select **FGS Template** and select the function configured in the previous step.
7. Set **Dependent Resource Type** to **bucket_name** and set the key of **Parameter** to **domain_id** and the value to the account ID.
8. Click **Save**.



Manually Remediating Resources

The following procedure shows how to manually configure remediation:

1. Go to the **Remediation Management** page.
2. On the **Resource Scope** tab, select target resources.
   - If you need to remediate the resources selected, click **Execute Remediation**.
   - If you do not need to remediate the resources, click **Add to Remediation Exception**.
3. Log in to the OBS console and go to the details page of the OBS bucket.
4. Check if the bucket policy has been modified.



FAQs

What Are the Differences Between Manual Remediation and Automatic Remediation?

Manual remediation requires you to manually search for and remediate noncompliant resources. Automatic remediation automatically applies remediation to noncompliant resources detected by a rule.

You are recommended to select manual remediation if it is the first time you configure remediation. Manual remediation can prevent service interruptions caused by resource modifications.

If you change manual remediation to automatic, all noncompliant resources detected after the change will be automatically remediated.

Why Does a Resource Fail to Be Remediated After the Remediation Is Applied?

This is typically caused by incorrect code in the FunctionGraph function or insufficient permissions of FunctionGraph. You can check the reasons by looking at the logs.

Why Is a Resource Still Noncompliant After the Remediation Is Applied?

Typically, a resource modification is reported to Config within 5 minutes of when the resource is modified, and the rule will be automatically triggered to generate the latest evaluation results.

