目录

云监控概览

使用云监控

用户可以使用云监控 API 对云监控服务进行相关操作，如读取监控数据，也可以调用相关的SDK进行读取监控数据。

术语表

名称	中文	说明
Namespace	命名空间	一个命名空间表示一类资源，在金山云中命名空间与云产品一一对应。KS3对象存储对应的命名空间为:"KS3"
Metric	指标	指标是监控的变量，监控数据是该变量随时间变化的数值。详见KS3监控项
Instance	实例	云资源实例，最小监控单元。KS3云监控的实例为Bucket，使用bucket name来表示实例ID
endpoint	云监控接入域名	云监控的服务接入地址：monitor.{region}.api.ksyun.com，{region}根据KS3 Bucket所在的Region不同而不同，详见KS3 Bucket Region和云监控endpoint对应表

KS3监控项

KS3云监控服务为客户提供计量方面的监控数据指标，借助云监控服务，客户可以实时的洞察KS3上资源使用情况。

监控指标	描述	单位
ks3.bucket.capacity.total.sd	标准存储量总量	byte
ks3.bucket.capacity.add.sd	标准存储量增量	byte
ks3.bucket.capacity.del.sd	标准存储量删除量	byte
ks3.bucket.capacity.total.ia	低频存储量总量	byte
ks3.bucket.capacity.add.ia	低频存储量增量	byte
ks3.bucket.capacity.del.ia	低频存储量删除量	byte
ks3.bucket.flow.down.sd	标准存储流量（下载）	byte
ks3.bucket.flow.down.ia	低频存储流量（下载）	byte
ks3.bucket.getcount.sd	标准get次数	次
ks3.bucket.putcount.sd	标准put次数	次
ks3.bucket.getcount.ia	低频get次数	次
ks3.bucket.putcount.ia	低频put次数	次
ks3.bucket.flow.up.ia	低频存储取回量	byte

KS3 Bucket Region和云监控endpoint对应表

KS3会将不同Region下的Bucket的监控数据，发到对应的云监控服务地址。用户在查询某个Bucket的监控数据时，需要查询Bucket所在的Region，然后去云监控对应的服务地址查询相关的监控项。

Region	云监控endpoint
北京（BEIJING）	monitor.cn-beijing-6.api.ksyun.com
上海（SHANGHAI）	monitor.cn-shanghai-2.api.ksyun.com
杭州（HANGZHOU）	monitor.cn-shanghai-2.api.ksyun.com
广州（GUANGZHOU）	monitor.cn-guangzhou-1.api.ksyun.com
香港（HONGKONG）	monitor.cn-hongkong-2.api.ksyun.com
俄罗斯（RUSSIA）	monitor.eu-east-1.api.ksyun.com
新加坡（SINGAPORE）	monitor.ap-singapore-1.api.ksyun.com

粒度和延迟说明

KS3云监控指标的按照天粒度采集聚合的，延迟为10个小时左右。在第二天上午10点以后，可以查看到前一天的监控数据。

使用Python SDK访问云监控

目录

开发前准备

安装KS3 Python SDK

因为用户在查询某个Bucket的监控数据时，需要查询Bucket所在的Region，去云监控对应的服务地址查询相关的监控项，所以需要安装KS3 SDK。详见KS3 SDK for Python使用指南。

安装云监控Python SDK

详见云监控 SDK说明。

运行环境

适用于2.6、2.7的Python版本，目前不支持Python 3版本。

初始化配置

1、用户首先需要在金山云控制台申请安全凭证，安全凭证包括access_key_id和secret_access_key。access_key_id 是用于标识API调用者的身份，secret_access_key是用于加密签名字符串和服务器端验证签名字符串的密钥。secret_access_key 必须严格保管，避免泄露。获取安全凭证方法: https://docs.ksyun.com/directories/1600

2、通过文件配置及管理密钥，参考examples内示例：

所在位置: '/etc/kscore.cfg' 或 './.kscore.cfg' 或 'C:\kscore.cfg'

> 注意：使用相对路径时，需与运行目录保持一致。

```
[Credentials]
ks_access_key_id=<your ak>
ks_secret_access_key=<your sk>
```

> 注意：配置文件中ak和sk不要加引号

3、用户也可以在程序运行时配置：

```
from kscore.session import get_session
# 密钥
ACCESS_KEY_ID = "<your ak>"
SECRET_ACCESS_KEY = "<your sk>"

s = get_session()
client = s.create_client("monitor", "cn-beijing-6", use_ssl=True, ks_access_key_id=ACCESS_KEY_ID, ks_secre
t_access_key=SECRET_ACCESS_KEY)
```

Demo演示

```
from kscore.session import get_session
import json

if __name__ == "__main__":
    s = get_session()

    # 不同Region的bucket需要设置对应的参数去查询
    # 北京region cn-beijing-6
    # 上海region cn-shanghai-2
    # 杭州region cn-shanghai-2
    # 香港region cn-hongkong-2
    # 俄罗斯region eu-east-1

    #GetMetricStatistics
    client = s.create_client("monitor", "cn-beijing-6", use_ssl=True)

    #获取一天的标准存储量总量
    m=client.get_metric_statistics(InstanceID="<YourBucketName>",Namespace="KS3",MetricName="ks3.bucket.ca
pacity.total.sd",StartTime="2018-07-03T00:00:05Z",EndTime="2018-07-04T00:00:05Z",Period="86400",Aggregate=
"Max")
    print json.dumps(m,sort_keys=True,indent=4)

    #获取一天的标准存储量增量
    m=client.get_metric_statistics(InstanceID="<YourBucketName>",Namespace="KS3",MetricName="ks3.bucket.ca
pacity.add.sd",StartTime="2018-03-18T00:00:00Z",EndTime="2018-03-19T00:00:00Z",Period="86400",Aggregate="M
ax")
    print json.dumps(m,sort_keys=True,indent=4)
    # 获取一天的标准存储量删除量
    m = client.get_metric_statistics(InstanceID="<YourBucketName>", Namespace="KS3", MetricName="ks3.bucke
t.capacity.del.sd",StartTime="2018-03-18T00:00:00Z",EndTime="2018-03-19T00:00:00Z", Period="86400",Aggrega
te="Max")
    print json.dumps(m, sort_keys=True, indent=4)
    # 获取一天的低频存储量总量
    m = client.get_metric_statistics(InstanceID="<YourBucketName>", Namespace="KS3", MetricName="ks3.bucke
t.capacity.total.ia",StartTime="2018-03-18T00:00:00Z",EndTime="2018-03-19T00:00:00Z", Period="86400",Aggre
gate="Max")
    print json.dumps(m, sort_keys=True, indent=4)
    # 获取一天的低频存储量增量
```

```
    m = client.get_metric_statistics(InstanceID="<YourBucketName>", Namespace="KS3", MetricName="ks3.bucke
t.capacity.add.ia",StartTime="2018-03-18T00:00:00Z",EndTime="2018-03-19T00:00:00Z", Period="86400",Aggrega
te="Max")
    print json.dumps(m, sort_keys=True, indent=4)
    # 获取一天的低频存储量增量
    m = client.get_metric_statistics(InstanceID="<YourBucketName>", Namespace="KS3", MetricName="ks3.bucke
t.capacity.del.ia",StartTime="2018-03-18T00:00:00Z", EndTime="2018-03-19T00:00:00Z", Period="86400",Aggreg
ate="Max")
    print json.dumps(m, sort_keys=True, indent=4)
    # 获取一天的标准存储的下载流量
    m = client.get_metric_statistics(InstanceID="<YourBucketName>", Namespace="KS3", MetricName="ks3.bucke
t.flow.down.sd",StartTime="2018-03-18T00:00:00Z", EndTime="2018-03-19T00:00:00Z", Period="86400",Aggregate
="Max")
    print json.dumps(m, sort_keys=True, indent=4)
    # 获取一天的低频存储的下载流量
    m = client.get_metric_statistics(InstanceID="<YourBucketName>", Namespace="KS3", MetricName="ks3.bucke
t.flow.down.ia",StartTime="2018-03-18T00:00:00Z", EndTime="2018-03-19T00:00:00Z", Period="86400",Aggregate
="Max")
    print json.dumps(m, sort_keys=True, indent=4)
    # 获取一天的标准存储的get请求数
    m = client.get_metric_statistics(InstanceID="<YourBucketName>", Namespace="KS3", MetricName="ks3.bucke
t.getcount.sd",StartTime="2018-03-18T00:00:00Z", EndTime="2018-03-19T00:00:00Z", Period="86400",Aggregate=
"Max")
    print json.dumps(m, sort_keys=True, indent=4)
    # 获取一天的标准存储的put请求数
    m = client.get_metric_statistics(InstanceID="<YourBucketName>", Namespace="KS3", MetricName="ks3.bucke
t.putcount.sd",StartTime="2018-03-18T00:00:00Z", EndTime="2018-03-19T00:00:00Z", Period="86400",Aggregate=
"Max")
    print json.dumps(m, sort_keys=True, indent=4)
    # 获取一天的低频存储的get请求数
    m = client.get_metric_statistics(InstanceID="<YourBucketName>", Namespace="KS3", MetricName="ks3.bucke
t.getcount.ia",StartTime="2018-03-18T00:00:00Z", EndTime="2018-03-19T00:00:00Z", Period="86400",Aggregate=
"Max")
    print json.dumps(m, sort_keys=True, indent=4)
    # 获取一天的低频存储的put请求数
    m = client.get_metric_statistics(InstanceID="<YourBucketName>", Namespace="KS3", MetricName="ks3.bucke
t.flow.down.sd",StartTime="2018-03-18T00:00:00Z", EndTime="2018-03-19T00:00:00Z", Period="86400",Aggregate
="Max")
    print json.dumps(m, sort_keys=True, indent=4)
    # 获取一天的低频存储的数据取回量
    m = client.get_metric_statistics(InstanceID="<YourBucketName>", Namespace="KS3", MetricName="ks3.bucke
t.flow.down.sd",StartTime="2018-03-18T00:00:00Z", EndTime="2018-03-19T00:00:00Z", Period="86400",Aggregate
="Max")
    print json.dumps(m, sort_keys=True, indent=4)
```

使用Java SDK访问云监控

目录

开发前准备

安装KS3 Java SDK

因为用户在查询某个Bucket的监控数据时，需要查询Bucket所在的Region，去云监控对应的服务地址查询相关的监控项，所以需要安装KS3 SDK。详见KS3 SDK For Java使用指南。

安装云监控Python SDK

详见云监控 SDK说明。

运行环境

适用于Java 5 以上开发环境

Demo演示

```
import org.apache.log4j.Logger;

import com.ksc.auth.AWSCredentials;
import com.ksc.auth.BasicAWSCredentials;
import com.ksc.monitor.KSCMonitorClient;
import com.ksc.monitor.model.GetMetricStatisticsRequest;
import com.ksc.monitor.model.GetMetricStatisticsResponse;

/**
 * Created by cutedandan on 2017/7/3.
 */
public class ks3Size {
    private static final Logger log = Logger.getLogger(ks3Size.class);

    public static void main(String[] args){
        String ak = "<your ak>";
        String sk = "<your sk>";

        AWSCredentials credentials = new BasicAWSCredentials(ak, sk);
        KSCMonitorClient client = new KSCMonitorClient(credentials);

        # 不同Region的bucket需要设置对应的参数去查询
        # 北京region monitor.cn-beijing-6.api.ksyun.com
        # 上海region monitor.cn-shanghai-2.api.ksyun.com
        # 杭州region monitor.cn-shanghai-2.api.ksyun.com
        # 香港region monitor.cn-hongkong-2.api.ksyun.com
        # 俄罗斯region monitor.eu-east-1.api.ksyun.com
        client.setEndpoint("monitor.cn-beijing-6.api.ksyun.com");

        String bucket="<YourBucketName>";
        String startTime="2017-09-04T016:00:00Z";
        String endTime="2017-09-05T016:00:00Z";
        //获取标准存储的总量
        GetMetricStatisticsRequest request=new GetMetricStatisticsRequest();
        request.setNamespace("KS3");      //固定设置为KS3
        request.setInstanceId(bucket);   // 设置对应要查询的bucket name
        request.setMetricName("ks3.bucket.capacity.total.sd");  //要查询的参数
        request.setStartTime(startTime);   // 设置起始和截止时间为查询7月2日的数据
        request.setEndTime(endTime);
        request.setPeriod(24*60*60);
        request.setAggregate("Max");
        request.setVersion("2010-05-25");

        GetMetricStatisticsResponse result = client.getMetricStatistics(request);
        String lable = result.getGetMetricStatisticsResult().getLabel();

        System.out.println("result: " + result);
```

```
        System.out.println("result: " + result);
        System.out.println("lable: " + lable);
        System.out.println("Datapoints: " + result.getGetMetricStatisticsResult().getDatapoints().get(0).g
etMax());

        // 获取一天中标准存储的增量
        request = new GetMetricStatisticsRequest();
        request.setNamespace("KS3");
        request.setInstanceId(bucket);
        request.setMetricName("ks3.bucket.capacity.add.sd");
        request.setStartTime(startTime);
        request.setEndTime(endTime);
        request.setPeriod(24*60*60);
        request.setAggregate("Max");
        request.setVersion("2010-05-25");

        result= client.getMetricStatistics(request);
        lable = result.getGetMetricStatisticsResult().getLabel();

        System.out.println("result: " + result);
        System.out.println("lable: " + lable);
        System.out.println("Datapoints: " + result.getGetMetricStatisticsResult().getDatapoints().get(0).g
etMax());

        //获取一天中标准存储的删除量
        request = new GetMetricStatisticsRequest();
        request.setNamespace("KS3");
        request.setInstanceId(bucket);
        request.setMetricName("ks3.bucket.capacity.del.sd");
        request.setStartTime(startTime);
        request.setEndTime(endTime);
        request.setPeriod(24*60*60);
        request.setAggregate("Max");
        request.setVersion("2010-05-25");

        result= client.getMetricStatistics(request);
        lable = result.getGetMetricStatisticsResult().getLabel();

        System.out.println("result: " + result);
        System.out.println("lable: " + lable);
        System.out.println("Datapoints: " + result.getGetMetricStatisticsResult().getDatapoints().get(0).g
etMax());

        System.out.println(result.getGetMetricStatisticsResult().getDatapoints().get(0).getTimestamp());
    }
}
```

使用 GO SDK访问云监控

目录

开发前准备

安装KS3 GO SDK

因为用户在查询某个Bucket的监控数据时，需要查询Bucket所在的Region，去云监控对应的服务地址查询相关的监控项，所以需要安装KS3 SDK。详见KS3 SDK for GO使用指南。

安装云监控 GO SDK

详见云监控 SDK说明。

运行环境

基于aws-sdk-go改造，适用于golang 1.8 开发环境。

初始化配置

1、用户首先需要在金山云控制台申请安全凭证，安全凭证包括access_key_id和secret_access_key。access_key_id 是用于标识API调用者的身份，secret_access_key是用于加密签名字符串和服务器端验证签名字符串的密钥。secret_access_key 必须严格保管，避免泄露。获取安全凭证方法：https://docs.ksyun.com/directories/1600

2、将获取到的access_key_id和secret_access_key（AK和SK）在程序进行配置。

Demo演示

此Demo演示了：如果获取某一账号下所有bukcet的标准存储量。

```
package main

import (
    "encoding/json"
    "fmt"
    "github.com/ks3sdklib/aws-sdk-go/aws"
    "github.com/ks3sdklib/aws-sdk-go/aws/credentials"
    "github.com/ks3sdklib/aws-sdk-go/service/s3"
    "github.com/ksc/ksc-sdk-go/ksc"
    "github.com/ksc/ksc-sdk-go/ksc/utils"
    "github.com/ksc/ksc-sdk-go/service/monitor"
)

func query_monitor(bucket string, region string, ak string, sk string) {
    //debug模式的话 打开这个开关
    svc := monitor.SdkNew(ksc.NewClient(ak, sk /*true*/), &ksc.Config{Region: &region}, &utils.UrlInfo{
        UseSSL: true,
    })
    //****************************************获取监控项(ListMetrics)************************************
*************
    m_metrics := make(map[string]interface{})
    m_metrics["Namespace"] = "KS3"
    m_metrics["InstanceID"] = bucket
    m_metrics["MetricName"] = "ks3.bucket.capacity.total.sd"
    m_metrics["StartTime"] = "2019-06-22T00:00:00Z"
    m_metrics["EndTime"] = "2019-06-23T00:00:00Z"
    m_metrics["Period"] = "86400"
    m_metrics["Aggregate"] ="Max,min,avg"
    resp, err := svc.GetMetricStatistics(&m_metrics)

    if err != nil {
        fmt.Println(err.Error())
    }
    if resp != nil {
```

```
    if resp != nil {
        str, _ := json.Marshal(&resp)
        fmt.Printf("%+v\n", string(str))
    }
}

func main() {
    ak := "your_ak"
    sk := "your_sk"

    region_map := map[string]string {
        "BEIJING": "cn-beijing-6",
        "SHANGHAI": "cn-shanghai-2",
        "HANGZHOU": "cn-shanghai-2",
        "GUANGZHOU": "cn-guangzhou-1",
        "HONGKONG": "cn-hongkong-2",
        "RUSSIA": "eu-east-1",
        "SINGAPORE": "ap-singapore-1",
    }

    var cre = credentials.NewStaticCredentials(ak, sk,"")
    var svcKs3 = s3.New(&aws.Config{
        Region: "HANGZHOU",
        Credentials: cre,
        Endpoint:"kss.ksyun.com",
        DisableSSL:true,
        LogLevel:1,
        S3ForcePathStyle:true,
        LogHTTPBody:true,
    })

    out,err := svcKs3.ListBuckets(nil)
    if err != nil {
        fmt.Println(err)
        return
    }

    buckets := out.Buckets
    for i:=0;i<len(buckets);i++{
        fmt.Println(*buckets[i].Name)
        fmt.Println(*buckets[i].Region)
        query_monitor(*buckets[i].Name, region_map[*buckets[i].Region], ak, sk)
    }

}
```