



how to add qwencloud and tencentcloud GPU usage to my current multi-cloud provisioning software

The cleanest approach is to give each cloud its own provider adapter that maps onto your existing GPU and usage model. One caveat on naming: Qwen is Alibaba's model family, not a separate GPU cloud. "QwenCloud" GPU usage can mean either Alibaba Cloud ECS GPU instances that you provision yourself, or calls to the hosted Qwen API, which are billed by tokens. You'll probably want to support both, as two separate kinds of resource.

What each provider exposes

Concern	Tencent Cloud	Alibaba Cloud (Qwen)
SDK	tencentcloud-sdk-python. You can install only <code>-COMMON</code> and <code>-CVM</code> , or use Common Client mode to call any product ^[1]	alibabacloud_ecs20140526 (ECS SDK V2.0) ^[2] ^[3]
Credentials	SecretId and SecretKey ^[1]	AccessKey pair from a RAM user that has only the permissions it needs ^[2]
Endpoint	Per product, e.g. <code>cvm.tencentcloudapi.com</code> ^[1]	Per region, e.g. <code>region_id='cn-hangzhou'</code> ^[2]
Discovering GPU instance types	CVM instance type APIs; filter on GPU families (GN-series)	DescribeInstanceTypeFamilies and instance-type APIs; filter on <code>gn*</code> families ^[3]
Inventory	CVM DescribeInstances ^[4]	DescribeInstances ^[2]
Hosted model usage	n/a	Qwen API is OpenAI-compatible, with Chat, Realtime and Batch APIs ^[5]
Terraform	tencentcloudstack/tencentcloud	aliyun/alibabacloud

If you use Apache Libcloud, note that it has an Aliyun ECS driver but, as far as I know, no maintained Tencent driver. For Tencent you'll need to write a native SDK adapter either way.

Suggested integration design

1. **Make GPUs first-class in your model.** Define a `GpuOffering` (provider, region, zone, instance type, GPU model, GPU count, vRAM, price per hour, spot available) and a `GpuUsageRecord` (resource ID, start and end time, GPU-hours, cost, currency, tags, project).
2. **Add one adapter per provider behind your existing interface:** `list_gpu_offerings()`, `provision()`, `terminate()`, `list_instances()`, `fetch_usage(start, end)`.

3. **Keep provisioning in Terraform**, using the two providers above. Use the SDKs only for discovery, checking whether types are in stock, and metering.
4. **Collect usage from three sources:**
 - **Inventory polling:** running GPU instances × elapsed hours gives near-real-time GPU-hours.
 - **Billing APIs:** Tencent billing and Alibaba BSS bill details give the authoritative cost. Reconcile against these every day.
 - **Monitoring:** Tencent Cloud Monitor and Alibaba CloudMonitor GPU metrics give utilization (idle-GPU alerts, right-sizing).
5. **Meter Qwen API usage separately.** Read the usage block (prompt and completion tokens) from each OpenAI-compatible response through your gateway. Store it as token usage, not GPU-hours.
6. **Fit it into your identity stack.** Store keys in Vault, ideally as short-lived STS/CAM role credentials rather than static keys. Gate actions like "provision a GPU on provider X" in OpenFGA.

Minimal adapter sketch

```
# Tencent
from tencentcloud.common import credential
from tencentcloud.cvm.v20170312 import cvm_client, models
def tc_gpu_instances(sid, skey, region):
    c = cvm_client.CvmClient(credential.Credential(sid, skey), region)
    r = models.DescribeInstancesRequest(); r.Limit = 100
    return [i for i in c.DescribeInstances(r).InstanceSet
            if i.InstanceType.startswith(("GN", "GI", "GT"))]

# Alibaba
from alibabacloud_ecs20140526.client import Client
from alibabacloud_ecs20140526 import models as m
from alibabacloud_tea_openapi.models import Config
def ali_gpu_instances(ak, sk, region):
    c = Client(Config(access_key_id=ak, access_key_secret=sk,
                     endpoint=f"ecs.{region}.aliyuncs.com"))
    res = c.describe_instances(m.DescribeInstancesRequest(region_id=region, page_size=100))
    return [i for i in res.body.instances.instance
            if i.instance_type.startswith("ecs.gn")]
```

Before you rely on these, check the instance-family prefixes (and the SDK's Python version support) in each console. Both clouds add new GPU families often, and their availability differs between international and China regions.

Gotchas

- Tencent China and international accounts use different SDKs and docs (`tencentcloud-sdk-python-intl-en` exists), and some GPU types only exist in China regions.^[4]
- GPU quotas are usually zero by default on both clouds, so build a quota-check step before provisioning.
- Bills come in different currencies (CNY or USD), so convert them in your cost layer.

If you tell me what your current stack is built on (Libcloud, Terraform, or a custom SDK layer), I can write the full adapter for it.

✱

1. <https://intl.cloud.tencent.com/document/product/583/19698>
2. <https://www.alibabacloud.com/help/doc-detail/93108.htm>
3. <https://www.alibabacloud.com/help/en/sdk/developer-reference/use-the-alibaba-cloud-python-sdk-through-the-ide>
4. https://github.com/TencentCloud/tencentcloud-sdk-python-intl-en/blob/master/tencentcloud/cvm/v20170312/cvm_client.py
5. <https://qwen.ai/apiplatform>
6. <https://www.alibabacloud.com/help/en/ecs/>
7. <https://pypi.org/project/tencentcloud-sdk-python-vod/>
8. <https://github.com/aliyun/alibabacloud-python-sdk-v2>
9. <https://www.spheron.network/blog/deploy-qwen-3-6-plus-gpu-cloud/>
10. <https://context7.com/aliyun/alibabacloud-python-sdk-v2>
11. <https://context7.com/tencentcloud/tencentcloud-sdk-python>
12. <https://www.tencentcloud.com/document/product/213/54596>
13. <https://intl.cloud.tencent.com/document/product/213/495>
14. https://staticintl.cloudcachetci.com/doc/pdf/product/pdf/tencent-cloud_1080_38753_en.pdf
15. <https://www.alibabacloud.com/help/en/ecs/developer-reference/faq-about-installing-alibaba-cloud-sdk-for-python-on-a-linux-ecs-instance>