



libcloud

With Libcloud you can wire Alibaba in using the built-in Aliyun ECS driver. Tencent Cloud has no Libcloud driver, so you'll write a small custom NodeDriver around the Tencent SDK and register it as a third-party driver. Libcloud doesn't handle GPU metrics, billing or Qwen API token usage, so those need a separate metering layer.

Where each cloud fits

Piece	Alibaba (Qwen)	Tencent
Libcloud driver	Built in: <code>Provider.ALIYUN_ECS</code> → <code>ECSDriver</code> [1]	None. Write a custom driver and register it [2]
Region scope	One driver instance per region (a single-region driver) [3]	Same pattern: one client per region
GPU discovery	<code>list_sizes()</code> , then filter <code>ecs.gn*</code>	<code>DescribeInstanceTypeConfigs</code> , then filter GPU families
Usage and billing	Outside Libcloud (BSS and CloudMonitor)	Outside Libcloud (Billing and Cloud Monitor)
Qwen API token usage	Outside Libcloud. The API is OpenAI-compatible [4]	n/a

Alibaba via ECSDriver

```
from libcloud.compute.types import Provider
from libcloud.compute.providers import get_driver

ECS = get_driver(Provider.ALIYUN_ECS)
drv = ECS(ak, sk, region="ap-southeast-1")    # Singapore

gpu_sizes = [s for s in drv.list_sizes() if s.id.startswith("ecs.gn")]
size = next(s for s in gpu_sizes if s.id == "ecs.gn7i-c8g1.2xlarge")
image = next(i for i in drv.list_images() if "ubuntu_22" in i.id)

node = drv.create_node(
    name="gpu-worker-1", size=size, image=image,
    ex_security_group_id="sg-xxx", ex_vswitch_id="vsw-xxx",
    ex_internet_charge_type="PayByTraffic",
)
```

Two caveats:

- The driver's built-in size catalog can fall behind Alibaba's actual offerings. One example is the `InstanceType.Offline` error on `create_node`. Check the type is actually in stock (Alibaba's `DescribeAvailableResource` API) before provisioning. If the driver has no `NodeSize` for a new GPU type, build one yourself. ^[5]
- If the driver lacks a GPU feature you need, such as spot strategy or newer disk categories, subclass `ECSDriver` and add `ex_*` parameters. Don't fork `Libcloud`.

Tencent custom driver

```
from libcloud.compute.base import NodeDriver, Node, NodeSize, NodeImage
from libcloud.compute.types import NodeState
from tencentcloud.common import credential
from tencentcloud.cvm.v20170312 import cvm_client, models

STATE = {"RUNNING": NodeState.RUNNING, "STOPPED": NodeState.STOPPED,
         "PENDING": NodeState.PENDING, "TERMINATING": NodeState.TERMINATED}
GPU_PREFIX = ("GN", "GI", "GT", "PNV", "HCCPNV")

class TencentCVMNodeDriver(NodeDriver):
    type = "tencent_cvm"
    name = "Tencent Cloud CVM"
    website = "https://www.tencentcloud.com"

    def __init__(self, key, secret, region, **kw):
        self.region = region
        self.client = cvm_client.CvmClient(credential.Credential(key, secret), region)

    def list_sizes(self, location=None):
        r = models.DescribeInstanceTypeConfigsRequest()
        out = []
        for c in self.client.DescribeInstanceTypeConfigs(r).InstanceTypeConfigSet:
            out.append(NodeSize(id=c.InstanceType, name=c.InstanceType,
                                ram=c.Memory * 1024, disk=None, bandwidth=None, price=None,
                                driver=self, extra={"cpu": c.CPU, "gpu": getattr(c, "GPU", 0),
                                                    "zone": c.Zone, "family": c.InstanceFamily}))
        return out

    def list_gpu_sizes(self):
        return [s for s in self.list_sizes()
                if s.extra["gpu"] or s.id.startswith(GPU_PREFIX)]

    def list_nodes(self):
        r = models.DescribeInstancesRequest(); r.Limit = 100; r.Offset = 0
        nodes = []
        while True:
            resp = self.client.DescribeInstances(r)
            nodes += [self._to_node(i) for i in resp.InstanceSet]
            r.Offset += 100
            if r.Offset >= resp.TotalCount: return nodes

    def create_node(self, name, size, image, ex_zone, ex_subnet_id,
                    ex_vpc_id, ex_sg_ids=(), ex_spot=False, **kw):
        r = models.RunInstancesRequest()
        r.from_json_string(__import__("json").dumps({
```

```

        "InstanceType": size.id, "ImageId": image.id, "InstanceName": name,
        "Placement": {"Zone": ex_zone},
        "VirtualPrivateCloud": {"VpcId": ex_vpc_id, "SubnetId": ex_subnet_id},
        "SecurityGroupIds": list(ex_sg_ids),
        "InstanceChargeType": "SPOTPAID" if ex_spot else "POSTPAID_BY_HOUR",
    )))
    iid = self.client.RunInstances(r).InstanceIdSet[0]
    return Node(id=iid, name=name, state=NodeState.PENDING,
                public_ips=[], private_ips=[], driver=self,
                size=size, image=image)

def destroy_node(self, node):
    r = models.TerminateInstancesRequest(); r.InstanceIds = [node.id]
    self.client.TerminateInstances(r); return True

def _to_node(self, i):
    return Node(id=i.InstanceId, name=i.InstanceName,
                state=STATE.get(i.InstanceState, NodeState.UNKNOWN),
                public_ips=i.PublicIpAddresses or [], private_ips=i.PrivateIpAddresses or [],
                driver=self, extra={"instance_type": i.InstanceType,
                                    "zone": i.Placement.Zone, "created": i.CreatedTime,
                                    "charge_type": i.InstanceChargeType})

```

Register it so the rest of your platform reaches it through the same `get_driver` call as every other provider:

```

from libcloud.compute.providers import set_driver, get_driver
set_driver("tencent_cvm", "yourpkg.drivers.tencent", "TencentCVMNodeDriver")
drv = get_driver("tencent_cvm")(sid, skey, region="ap-singapore")

```

Check the GPU family prefixes and the GPU field on `InstanceTypeConfig` against your account's actual output. Tencent's GPU catalog varies by region, and China and international regions differ.

Metering layer alongside Libcloud

Libcloud only exposes Node lifecycle, so add a `GpuUsageCollector` interface with one implementation per provider:

1. Poll `list_nodes()` on a schedule. For GPU nodes, record running intervals $\times$ GPU count to get GPU-hours. This works the same for Alibaba, Tencent and your existing providers.
2. Reconcile cost against each provider's billing API once a day (Alibaba BSS bill details, Tencent Billing bill details), matched by instance ID.
3. Optionally pull GPU utilization from CloudMonitor or Cloud Monitor to flag idle GPUs.
4. Meter Qwen API usage at your gateway by logging the usage token counts from each response. Store them as token usage, not GPU-hours.

Keep the AK/SK and SecretId/SecretKey pairs in Vault, and give each driver short-lived credentials when you create it.

1. https://libcloud.readthedocs.io/en/stable/compute/drivers/aliyun_ecs.html
2. https://docs.libcloud.apache.org/en/stable/third_party_drivers.html
3. https://libcloud.readthedocs.io/en/stable/supported_providers.html
4. <https://qwen.ai/apiplatform>
5. <https://github.com/apache/libcloud/issues/1689>
6. https://www.mdjonline.com/eedition/segment-3-cutout/cutout_9c73e201-807b-5dc5-8fcc-60de45587792.html?mode=print
7. <https://technode.com/?s=alibaba>
8. <https://news.futunn.com/en/post/1000203653/cicc-insights-into-the-development-of-productivity-agents-from-workbuddy>
9. <http://epaper.deccanchronicle.com/articledetailpage.aspx?id=20065414>
10. <https://gamingonphone.com/category/nft/>
11. <https://www.tencentcloud.com/document/product/1155/62021>
12. <https://cloud.tencent.com/developer/information/libcloud>
13. https://www.alibabacloud.com/en/product/ecs?_p_lc=1
14. <https://www.aliyun.com/>
15. <https://libcloud.apache.org/>
16. https://libcloud.readthedocs.io/en/latest/compute/drivers/aliyun_ecs.html
17. <https://libcloud.apache.org/blog/2017/04/25/libcloud-2-0-0-released.html>
18. <https://libcloud.apache.org/getting-started.html>
19. <https://lists.apache.org/thread/mj8ylond9zghgknn21q4rkcrb9mmn9j7>
20. <https://libcloud.readthedocs.io/en/stable/loadbalancer/drivers/>
21. <https://github.com/AliyunContainerService/docker-machine-driver-aliyunecs>
22. <https://simplywall.st/stocks/hk/media/hkg-700/tencent-holdings-shares>
23. <https://substack.com/profile/171154836-cynthia/note/c-345872341>
24. <https://www.marktechpost.com/category/technology/open-source/page/70/>
25. <https://www.opensourceforu.com/category/etc/page/113/>