OCEANBASE

FROM INTRODUCTION TO PRACTICE

FROM INTRODUCTION TO PRACTICE

Lesson 2: Installation and Deployment of OceanBase Community Edition

Peng Wang

OceanBase Global Technical Evangelist

Agenda

- Preparation Before Deployment
- Introduction To Deploying Related Ecological Components
- Deploy Personal Experimental Environment
- Deploy Production Environment
- Introduction To Basic Cluster Usage

Preparation Before Deployment

Preparation of installation package

Download Path	Address
official website	https://en.oceanbase.com/softwarecenter
Github	https://github.com/oceanbase/oceanbase/releases
Official yum repository	https://mirrors.oceanbase.com/

Software Package	Software Package Prefix	Explanation
OBD Install Package	ob-deploy	Independent package body. You can download the version package as needed
OceanBase	oceanbase-ce (Database Package) oceanbase-ce-libs (Database Dependency) oceanbase-ce-utils (Database tool integration package)	Independent package body. You can download the package as needed
OceanBase Proxy	obproxy-ce	Independent package body. You can download the version package as needed
OceanBase One-click Installation Package	oceanbase-all-in-one	Includes database software and OBD, OBProxy, OBClient, OCP Express (starting from V4.1), Prometheus, and Grafana.
OCP Cloud Platform	ocp-all-in-one	Includes database software and OBD, OBProxy, and OCP

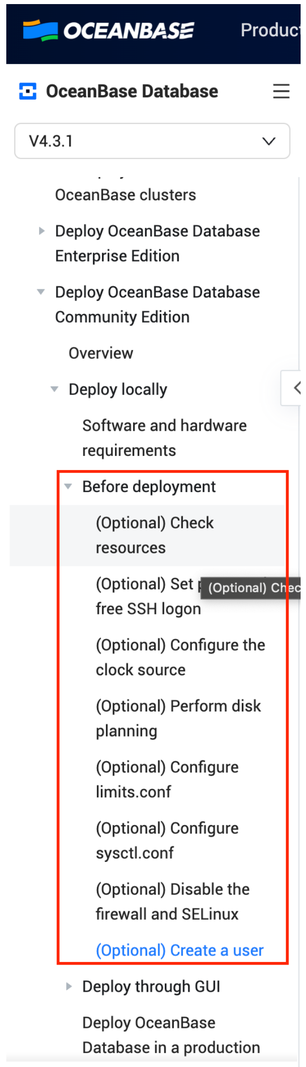
Directions:

1. If you use OBD deployment, it is recommended to download the oceanbase-all-in-one package directly for offline deployment
2. If you need to deploy the OCP cloud platform, it is recommended to download the ocp-all-in-one package directly for offline deployment

Preparation Before Deployment

Server initialization

1. Create users and groups (when NFS is used as a backup medium, uid/gid must be consistent across all nodes)
2. Configure user SSH trust (optional)
3. Configure sudo password free (OBD is not required, OCP is required)
4. Disk and file system planning (not necessary, in product ENV clog&data strongly recommends different disks)
5. Modify user resource restrictions (required for production)
6. Modify kernel parameters (required for production)
7. Turn off firewall and SELinux (adjust as needed)
8. Configure clock source (required for cluster environment)



For specific details, please refer to the official website's detailed introduction:
https://en.oceanbase.com/docs/common-oceanbase-database-10000000001378214

Scan the code to view the document

Agenda

- Preparation Before Deployment
- Introduction To Deploying Related Ecological Components
- Deploy Personal Experimental Environment
- Deploy Production Environment
- Introduction To Basic Cluster Usage

Introduction to Deploying Related Ecological Components

Compare Items	OBD	OCP	ob-operator
Supported installed vision	V3.1.x and V4.x	V3.1.x and V4.x	V4.1.x and above
Offline/Online Deployment	All supported	Only supports offline deployment	All supported
Whether to support self-compiled install packages	Supported	Unsupported	Unsupported
Whether the deployment tool supports high availability	Unsupported	Supported	Supported (Dependency on Kubernetes)
Whether to support single machine multi-node deployment	Supported	Unsupported	Supported
Ecological docking	Open source code	Open API	Open source code
Difficulty in getting started	Low	Medium	Depending on familiarity with the Kubernetes environment
Deployment tool resource usage	Lightweight	High	Lightweight

Introduction to Deploying Related Ecological Components

Comparison of function points	OBD	OCP	OB-Operator
Tenant creation	Supported	Supported	Supported
Tenant viewing	Supported	Supported	Supported
Database management	Unsupported	Supported	Not currently supported
User rights management	Unsupported	Supported	Not currently supported
resource management	Unsupported	Supported	Not currently supported
Merge management	Unsupported	Supported	Supported
Backup Recovery	Unsupported	Supported	Not currently supported
Monitoring alarms	Supported (dependent on Prometheus ecosystem)	Supported	Same as OBD
Scaling	Supports expansion, currently does not support scaling	Supported	Supported
Topsql/Slowsql	Unsupported	Supported	Not currently supported
Transaction diagnostics	Unsupported	Supported	Not currently supported
Deadlock diagnostics	Unsupported	Supported	Not currently supported
Session management	Unsupported	Supported	Not currently supported
Primary and secondary databases	Supported	Supported	Supported

Agenda

- Preparation Before Deployment
- Introduction To Deploying Related Ecological Components
- Deploy Personal Experimental Environment
- Deploy Production Environment
- Introduction To Basic Cluster Usage

Deploy Personal Experimental Environment

Demo Environment

A small-scale stand-alone environment that does not occupy many resources. It can be used for some basic functions but can NOT be used for production or performance testing. It focuses on quick deployment.

6G memory resources OB, OBProxy, OBAgent, Prometheus, Grafana will be installed at the same time. If you need to deploy specific components, refer to the quick deployment command. For example:
obd demo -c oceanbase-ce

Cluster environment

OCP-Express monitoring has relatively loose resource requirements. If resources are configured according to production requirements, it is also suitable for use in small cluster production environments.

OCP-Express has weak operation and maintenance capabilities

Small business volume, few cluster nodes, low reliance on GUI operation and maintenance

Container environment

Deploying a small stand-alone environment in a Docker environment is NOT suitable for production and performance testing. If you have relevant requirements, you can use ob-operator.

No promise yet Stability and performance of the deployment environment under Docker. Kuberates+OB-Operator can be deployed in a production environment

Agenda

- Preparation Before Deployment
- Introduction To Deploying Related Ecological Components
- Deploy Personal Experimental Environment
- Deploy Production Environment
- Introduction To Basic Cluster Usage

Deployment of Production Environment

High reliability deployment solution

Community Edition

- Three data centers and three replicas in the same city (**Figure 1**)
- Two locations and three centers "master-backup" deployment (**Figure 2**)
- Three locations, five centers and five replicas (**Figure 3**)
- Two data centers and "master-backup" deployment in the same city (**Figure 4**)

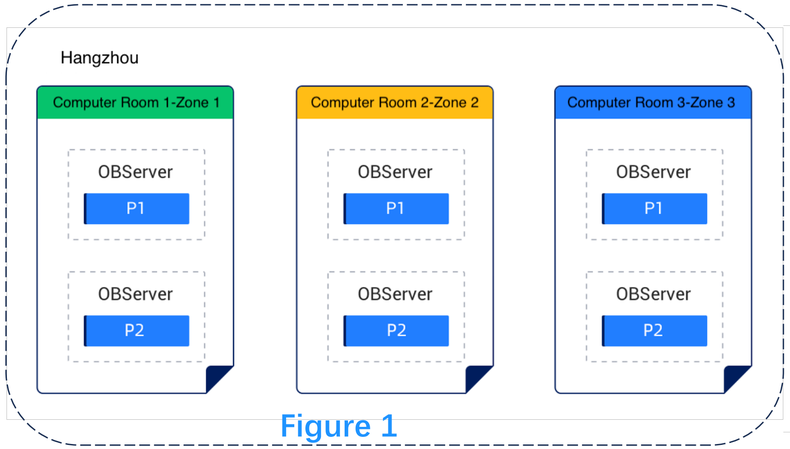


Figure 1

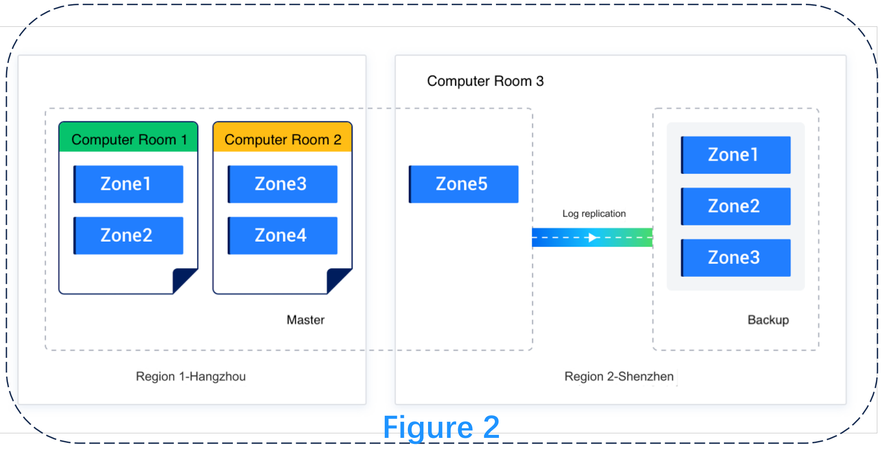


Figure 2

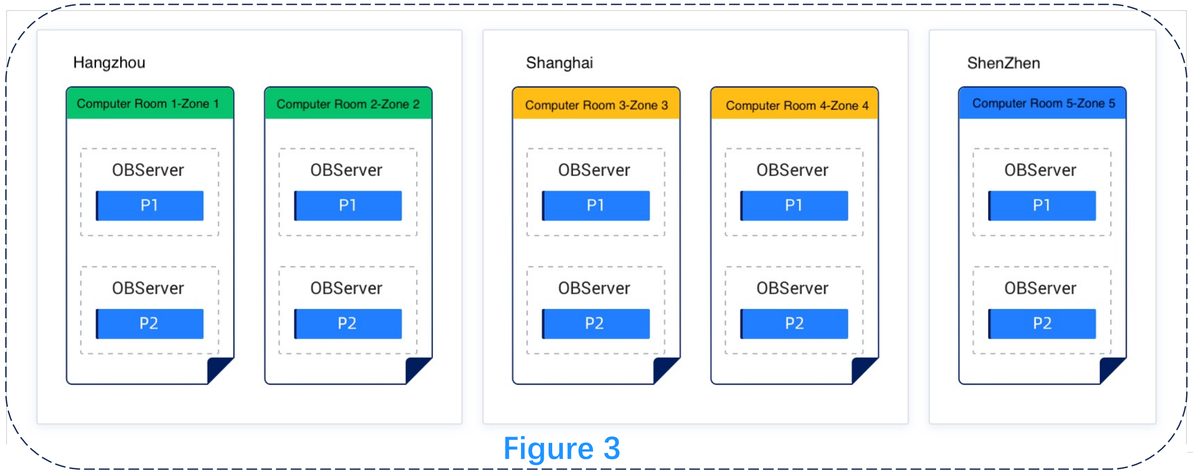


Figure 3

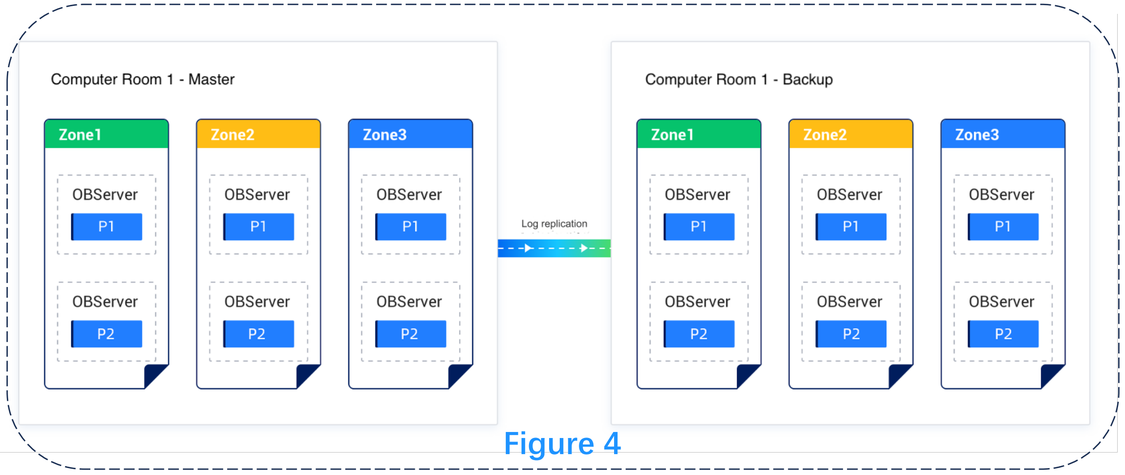


Figure 4

Deployment of Production Environment

Planning OceanBase cluster deployment

Planning and deployment requirements

Server Hardware and Software	• Support mainstream servers and domestically produced hardware and software • Support X86 and ARM architectures	**Server initialization requirements and command dependencies**
Resource Planning	• CPU: minimum 4C, recommended 32C • Memory: minimum 16G, recommended no less than 32G • Disk: SSD, XFS/EXT4 file system, capacity >= memory * 6	**Resource parameter calculation method**
User Planning	• sudo Permissions	**Recommended for ordinary users**
Directory Planning	• Installation directory, data directory, transaction log directory	**Directory partitioning, disk, capacity planning, etc.**

Deployment of Production Environment

Deploy a cluster through OCP

Why choose OCP

Differences between OCP-Express and OCP

OCP-Express: parameter management, tenant management, monitoring, and other functions. Meta tenants are in the business cluster and are suitable for cluster environments with <= 3 nodes.

It is recommended to use OCP operation and maintenance

OCP Common ARCH

Standalone and high-reliability

- Standalone: Dedicated server deployment of OCP+MetaDB
- High reliability: 3-node OCP+3-node MetaDB cluster

Mainly depends on high reliability requirements

What is Metadb

OceanBase + 2 meta-tenants

- MetaDB: Currently only supports OceanBase
- Meta-tenants: OCP metadata storage tenant + OCP monitoring data tenant

Not recommended to

- share meta-tenants.
- share meta-tenants and business clusters.

Install OCP

Deploy OCP Community Edition

OCP server resource requirements: 17C 60G

- **Note:** MetaDB disk resource usage.
- Graphical deployment is recommended.

Modules	CPU	Mem
OCP-Server	4	8G
MetaDB Database	13	52G
MetaDB System Tenant	5	28G
MetaDB Meta Tenant	4	8G
MetaDB Monitor Tenant	4	16G

OCP Deployment Cluster

1. Upload the software package
2. Add a host
3. Create a new cluster
4. Create a new service tenant
5. Create an OBProxy cluster

Precautions

Deployment of Production Environment

Deploy a cluster through OCP

STEP 01
Upload Package

1. Upload the required version installation package, which requires version matching;

2. OCP uploads the built-in database-related installation package by default;

STEP 02
Add Host

1. The host node needs to turn off the firewall and SELinux;
2. The host node needs to use a static IP;
3. The host node needs to create a credential operating system user in advance and configure sudo permissions;
4. The deployment (credential) user of the host node and OCP node needs to have clockdiff command operation permissions. If you do not have permissions, you can use the root user to execute the authorization setcap cap_net_raw+ep /usr/sbin/clockdiff.

STEP 03
Create a New Cluster

Check whether you need to customize the OceanBase system parameters, such as the number of system log files to be retained and the disk pre-occupied size.

STEP 04
New Business Tenant

It is forbidden to use the default sys tenant as a business tenant.

STEP 05
Create an OBProxy Cluster

1. It is recommended to back up the root@proxysys password. This password will be used if you need to take over OBProxy later.

2. It is recommended to deploy a 2-node OBProxy cluster at least. If the business volume is large, it is recommended to deploy the OBProxy cluster on a separate server.

Deployment of Production Environment

Deploy cluster via OBD

Common deployment architecture

The central control machine is recommended to be no less than 4C 8G. Components can be flexibly selected and deployment can be taken over.

4-Nodes Environment	Deployment Components	Description
Node A	OBD, OCP-Express	As a central control node, it is mainly used for OBD deployment management and operation and maintenance monitoring services. OCP-Express does not have a separate MetaDB and will create an OCP meta tenant in the business cluster.
Node B	OBServer, OBProxy, OBAgent	As OceanBase cluster node, OBProxy service node and OBAgent service node
Node C	OBServer, OBProxy, OBAgent	Same as above
Node D	OBServer, OBProxy, OBAgent	Same as above

Three ways to install OBD

RPM (public network recommended)
All-in-one (offline recommended)
Source code compilation

Precautions

Choose different deployment tools based on different requirements for operation and maintenance (OCP/OCP-Express)
Require java-1.8.0-openjdk (OBD270 supports automatic installation)

High-risk OBD commands

High-risk commands: destory (destroy), redeploy (reinstall cluster), reinstall (reinstall components)

Deployment of Production Environment

Common commands for deploying clusters through OBD

#View repository list
obd mirror list

#Disable remote repository access
obd mirror disable remote

#Enable remote repository access
obd mirror enable remote

#View the list of deployed services
obd cluster list

#View deployment service details
obd cluster display deployment_name

#Specify the configuration file to install the service
obd cluster deploy custom_deployment_name -c deployment_configuration_cile

#Destroy the deployment service, a high-risk operation that will clean up all data of the specified deployment service
obd cluster destroy deployment_name

#Edit or view a deployment configuration file
obd cluster edit-config deployment_name

#Start/Stop/Restart Services
obd cluster start/stop/restart deployment_name

#Start/stop/restart the specified service. The service name can be viewed through the ”obd cluster edit-config” module name
obd cluster start/stop/restart deployment_name -c service_name

#Start/stop/restart the specified service and node. The service name can be viewed through “obd cluster edit-config” command
obd cluster start/stop/restart deployment_name -c service_name -s IP1,IP2

#Reload configuration. After modifying the configuration through “obd cluster edit-config” command, a black screen will output an execution reminder
obd cluster reload deployment_name

#Create a single-machine OceanBase environment with minimal 2C 6G resources
obd demo

#Deploy the demo environment to specify the service name and port
obd demo -c oceanbase-ce --oceanbase-ce.mysql_port=3881 --oceanbase-ce.rpc_port=3882

#Create a business tenant to maximize the use of remaining resources
obd cluster tenant create deployment_name -n customize_tenant_name

#One-click performance test, currently supports mysqltest、sysbench、tpch、tpcc
obd test mysqltest/sysbench/tpch/tpcc

#One-click cluster inspection
obd obdiag check --cases= deployment_name

#Collect diagnostic information of the OceanBase cluster to which the deployment name belongs with one click
obd obdiag gather all deployment_name

#One-click diagnosis and analysis of the system logs of the OceanBase cluster to which the deployment name belongs during the specified time period
obd obdiag analyze log deployment_name --from 2024-02-06 18:00:00 --to 2024-02-06 18:10:00

#One-click trace log diagnosis of the OceanBase cluster to which the deployment name belongs
obd obdiag analyze flt_trace deployment_name

#One-click root cause analysis, supporting obproxy disconnection/cluster merge stuck/lock conflict diagnosis
obd obdiag rca --scene=disconnection/major_hold/lock_conflict

Deployment of Production Environment

Deploy OceanBase database through OB-Operator

Kubernetes Provides Resource Requirements

CPU：16
Memory：36G
Storage：200G

Kubernetes + OB-Operator Containerized Production Environment

cert-manager
local-path-provisioner

Deployment Process

Precautions

OB-Operator version 2.1.2 starts to support production environment deployment

OB-Operator versions before 2.2.0 or oceanbase versions before 4.3.0 will rely on the calico network plug-in

Agenda

- Preparation Before Deployment
- Introduction To Deploying Related Ecological Components
- Deploy Personal Experimental Environment
- Deploy Production Environment
- Introduction To Basic Cluster Usage

Introduction to Basic Cluster Usage

1.1 SQL 1 - View Cluster Resource Usage

```
select zone,svr_ip,svr_port,
cpu_capacity,cpu_assigned_max,cpu_capacity-
cpu_assigned_max as cpu_free,
round(memory_limit/1024/1024/1024,2) as memory_total_gb,
round((memory_limit-
mem_capacity)/1024/1024/1024,2) as system_memory_gb,
round(mem_assigned/1024/1024/1024,2) as mem_assigned_gb,
round((mem_capacity-mem_assigned)/1024/1024/1024,2) as memory_free_gb,
round(log_disk_capacity/1024/1024/1024,2) as log_disk_capacity_gb,
round(log_disk_assigned/1024/1024/1024,2) as log_disk_assigned_gb,
round((log_disk_capacity-
log_disk_assigned)/1024/1024/1024,2) as log_disk_free_gb,
round((data_disk_capacity/1024/1024/1024),2) as data_disk_gb,
round((data_disk_in_use/1024/1024/1024),2) as data_disk_used_gb,
round((data_disk_capacity-
data_disk_in_use)/1024/1024/1024,2) as data_disk_free_gb
from gv$ob_servers;
```

Before creating a user tenant, we need to confirm the resource usage of the current cluster. Here are two SQL statements and some instructions.

Using SQL 1, we can confirm the capacity of memory/cpu/disk on the corresponding observer node, the allocated and remaining capacity.

Note:
If you have just deployed the OceanBase database, you may find that the "data disk/log disk" will occupy a large amount after cluster deployment.

The reason is:

- The data disk size "data_file_size/datafile_disk_percentage" that controls the size of the data disk that OB can use is pre-occupied, and the corresponding parameters do not support reduction.
- The log disk size "log_disk_size/log_disk_percentage" that controls the size of the log disk used by OB logs is also pre-occupied, and the corresponding parameters can be increased or decreased as needed.

zone	svr_ip	svr_port	cpu_capacity	cpu_assigned_max	cpu_free	memory_total_gb	system_memory_gb	mem_assigned_gb	memory_free_gb	log_disk_capacity_gb	log_disk_assigned_gb	log_disk_free_gb	data_disk_gb	data_disk_used_gb	data_disk_free_gb
zone1	172.x.x.93	2882	16	2	14	12.00	5.00	1.00	6.00	50.00	3.00	47.00	50.00	0.06	49.94
zone2	172.x.x.94	2882	16	2	14	12.00	5.00	1.00	6.00	50.00	3.00	47.00	50.00	0.06	49.94
zone3	172.x.x.95	2882	16	2	14	12.00	5.00	1.00	6.00	50.00	3.00	47.00	50.00	0.06	49.94

3 rows in set (0.00 sec)

Introduction to Basic Cluster Usage

1.2 SQL 2 - View Cluster Resource Allocation Details

- If the tenant resources to be created are insufficient and there are no resources that can be expanded, you can view the specifications of existing tenants through SQL 2.
- Combined with monitoring, view the usage rate of corresponding resources and evaluate whether the specifications of existing tenants can be reduced before creating new user tenants.

```
select t4.tenant_id,t4.tenant_name,
t1.name resource_pool_name, t1.unit_count,
t2.`name` unit_config_name, t2.max_cpu, t2.min_cpu,
round(t2.memory_size/1024/1024/1024,2) mem_size_gb,
round(t2.log_disk_size/1024/1024/1024,2) log_disk_size_gb, t2.max_iops,
t2.min_iops, t3.unit_id, t3.zone, concat(t3.svr_ip,':',t3.`svr_port`) observer
from dba_ob_resource_pools t1
join dba_ob_unit_configs t2 on (t1.unit_config_id=t2.unit_config_id)
join dba_ob_units t3 on (t1.`resource_pool_id` = t3.`resource_pool_id`)
left join dba_ob_tenants t4 on (t1.tenant_id=t4.tenant_id)
order by t4.tenant_name,t3.zone;
```

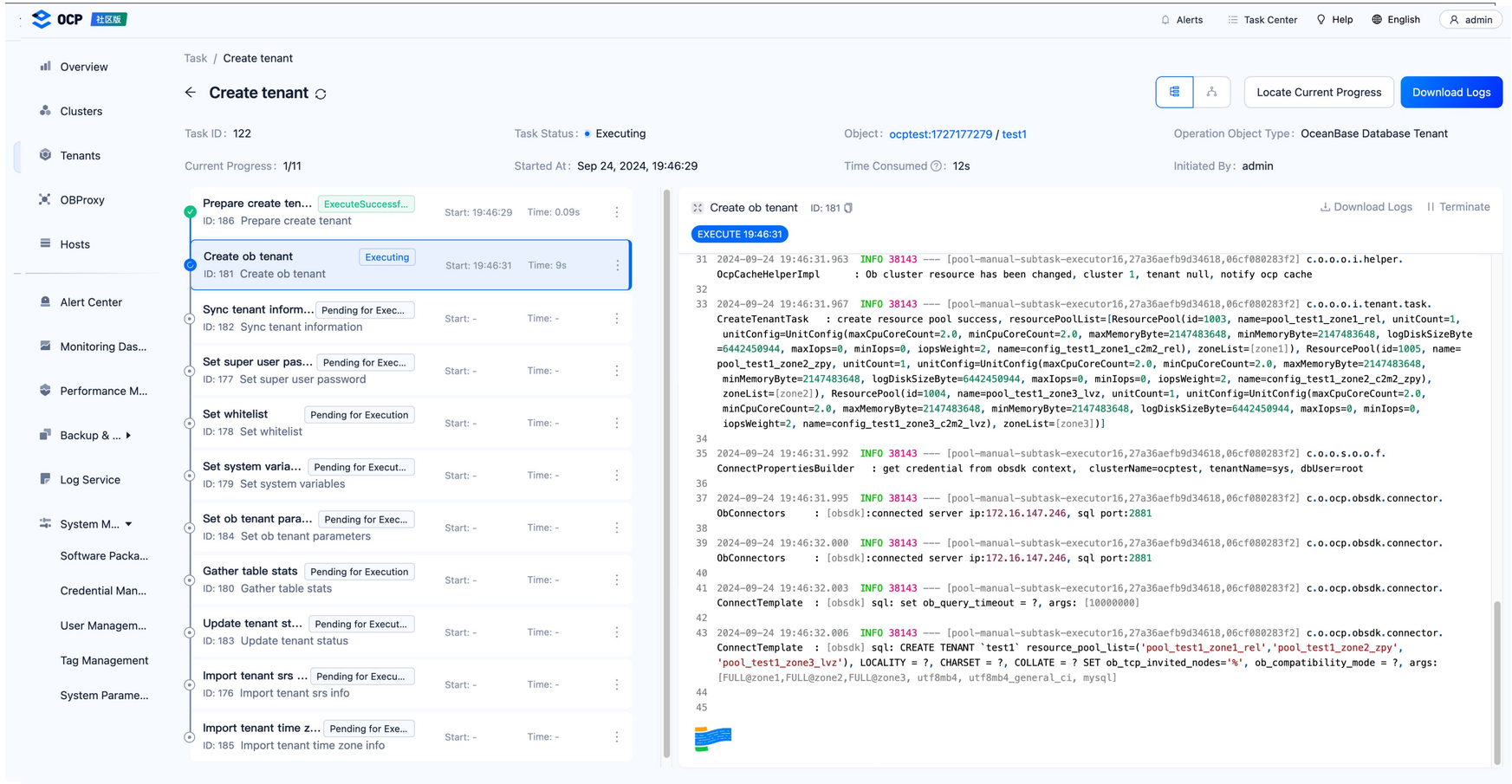
tenant_id	tenant_name	resource_pool_name	unit_count	unit_config_name	max_cpu	min_cpu	mem_size_gb	log_disk_size_gb	max_iops	min_iops	unit_id	zone	observer
1	sys	sys_pool	1	sys_unit_config	2	2	1.00	3.00	9223372036854775807	9223372036854775807	1	zone1	172.x.x.93:2882
1	sys	sys_pool	1	sys_unit_config	2	2	1.00	3.00	9223372036854775807	9223372036854775807	2	zone2	172.x.x.94:2882
1	sys	sys_pool	1	sys_unit_config	2	2	1.00	3.00	9223372036854775807	9223372036854775807	3	zone3	172.x.x.95:2882

3 rows in set (0.02 sec)

Introduction to Basic Cluster Usage

2. Create a user tenant for the MySQL schema

2.1 Create a tenant using OCP



Introduction to Basic Cluster Usage

2. Create a user tenant for the MySQL schema

2.2. Creating a tenant using OBD

```
# Create a tenant
obd cluster tenant create ob422 -n test2 --max-cpu=2 --
memory-size=2G --log-disk-size=6G --max-iops=10000 --unit-
num=1 --charset=utf8 -s 'ob_tcp_invited_nodes="%"'
```

```
# View Tenants
obd cluster tenant show ob422 -t test2
```

Note:

- It is recommended that the latest version of OBD be used.
- Here, **ob422** is the name corresponding to the OBD cluster list, deploy_name.
- **test2** is the name of the tenant created in the test example.
- The resource unit name used by the tenant created using the OBD command is **${tenant_name}_unit**, and the resource pool name is **${tenant_name}_pool**.
- The root user password of the tenant is empty by default. See the subsequent adjustment of creating a tenant using the SQL command line.

Introduction to Basic Cluster Usage

2. Create a user tenant for the MySQL schema

2.3. Create a tenant using the SQL command line

```
create resource unit u1 min_cpu=4,max_cpu=4,memory_size='4g',log_disk_size='12g',max_iops=10000;

create resource pool p1 unit='u1',zone_list=('zone1','zone2','zone3'),unit_num=2;

create tenant test1 resource_pool_list=('p1'),primary_zone='zone1,zone2,zone3',charset=utf8mb4,collate=utf8mb4_bin
set ob_tcp_invited_nodes='%';
```

```
create resource unit u2 min_cpu=4,max_cpu=4,memory_size='4g',log_disk_size='12g',max_iops=10000;

create resource pool p2_1 unit='u1',zone_list=('zone1'),unit_num=2;
create resource pool p2_2 unit='u1',zone_list=('zone2'),unit_num=2;
create resource pool p2_3 unit='u1',zone_list=('zone3'),unit_num=2;

create tenant test2 resource_pool_list=('p2_1','p2_2','p2_3'),primary_zone='zone1,zone2,zone3',charset=utf8mb4,collate=utf8mb4_bin
set ob_tcp_invited_nodes='%';
```

Creating a tenant consists of three steps:

1. Create resource unit specifications: This step is optional. If you have suitable specifications, you can skip this step and reuse directly.

2. Create a resource pool: You can have one resource pool for each zone, using independent resource unit specifications (OCP uses this method), or all zones can use the same resource unit specifications, all under one resource pool (OBD uses this method).

3. Create a tenant: When creating a tenant, you need to associate it with the resource pool created in step 2.

Note:

- By default, after the tenant is created, the root user password corresponding to the tenant is empty. In the production environment, it is recommended to adjust the password by this command in time and log in to the corresponding user tenant:

```
alter user root identified by ‘xxx’; or set password for root = password(‘xxxx’);
```

Introduction to Basic Cluster Usage

3. Connect tenants — Tenant connection example

```
obclient/mysql -h node_IP -P port -u username@tenant_name#cluster_name -p password -D database_name -A -c
```

Option	Description
-h	Provide the IP address of the OBServer or OBProxy to be accessed
-u	Provide the tenant's connection account, in the format of username@tenant_name#cluster_name or cluster_name:tenant_name:user_name. If the tenant name is not specified, the default is the sys tenant
-P	The default port is 2881 when connecting directly to the OBServer, and the default port is 2883 when connecting to the OBProxy. The specific port is subject to the actual environment
-p	Provide the account password. For security reasons, you can not provide it. Instead, enter it in the following prompt. The password text is not visible
-c	Indicates not to ignore comments in the MYSQL/OBClient running environment, such as hint comments
-D	Indicates the default database to be accessed after connection
-A	Indicates ignoring pre-read metadata information

Introduction to Basic Cluster Usage

3. Connect tenants — Tenant connection notes

General notes on connecting to tenants:

- When connecting directly to the OBServer (default port 2881), the cluster name cannot be included.
- About OBProxy connection (default port 2883). If there are multiple OceanBase clusters associated, the cluster name is required.
  - If OBProxy is started through **rs list**, you can specify the cluster name or not. You can confirm the rootservice_list parameter corresponding to OBproxy
  - If OBProxy is started via **configUrl**, the cluster name must be specified. You can confirm the obproxy_config_server_url parameter corresponding to OBProxy.

Note:
When deploying OBProxy, there are two deployment methods:

- **RsList deployment method:** when starting OBProxy, you can specify the RootServer information of the OceanBase cluster by specifying the **-r** parameter in the command
- **ConfigUrl deployment method:** indicates how to query the RootServer information of the OceanBase cluster by specifying the **obproxy_config_server_url** parameter item in the command when starting OBProxy

What is rs list:
The IP list of the machines with Root Service started in the OceanBase cluster (usually one per Zone)

Introduction to Basic Cluster Usage

4. Set cluster/tenant parameters (or variables)

4.1 Viewing and modifying parameters

Viewing Parameters

View on OCP:

1. Log in to OCP
2. Select Cluster in the left navigation bar to enter the cluster page
3. In the Cluster List area, select the cluster to be operated and click its cluster name
4. In the left navigation bar of the displayed page, click Parameter Management

View using OBD:

```
obd cluster edit-config ${deploy_name}
```

Note: The OBD command can only view the database parameters configured when using OBD deployment

View using SQL command line:

```
show parameters like ‘%memory_limit%’;
select * from gv$ob_parameters where name like '%memstore%';
```

Modifying Parameters

Modify parameters using SQL command line:

```
alter system set parameter='Parameter Value' [tenant='xxx'];
```

Modifying and viewing parameters on OBD and OCP are similar, but it should be noted that:

1. If the environment is deployed in OCP, whether you adjust the parameters directly on OCP or the SQL command line, both OCP and the command line will show the adjusted values.
2. If it is an OBD deployment environment, after modifying it through **edit-config**, the modified content can be seen normally in the command line. On the contrary, if you adjust the parameters through the SQL command line first, and then view it through edit-config, it is still the original configuration before the modification. At this time, if you forget to add **–wop** when you stop and start OBD, the original modified configuration will be lost
3. You can use the **find ~/.obd –name “parameter.yaml”** to confirm which parameters need to be reinstalled or restarted to take effect after modification.

Introduction to Basic Cluster Usage

4. Set cluster/tenant parameters (or variables)

4.2 Viewing and modifying variables

Viewing Variables

View variables on OCP

1. Log in to OCP
2. Click **Tenants** in the left navigation bar and click the tenant name in the tenant list
3. On the left navigation bar of the displayed page, click **Parameter Management**
4. On the parameter list page, you can view information about all parameters of the current cluster
   - The displayed information includes the parameter name, value type, value range, default value, current value, description, and whether it is read-only
   - The default value indicates the default value for a newly created tenant. Read-only means that the parameter cannot be modified, for example: lower_case_table_names is read-only and only takes effect when a tenant is created. If it is a modifiable parameter, a Modify Value button will be displayed in the Operation column
   - Check the upper right corner to view only modified parameters to view the list of parameters that have been modified in OCP

View variables using the SQL command line

```
show variables like 'ob_query_timeout’;
select * from CDB_OB_SYS_VARIABLES where name='ob_query_timeout' and tenant_id=x;
select * from DBA_OB_SYS_VARIABLES where name='ob_query_timeout';
```

Modifying Variables

OCP modifies variables and views them **on the same page**. For details, please refer to the steps for viewing variables

Modify variables using SQL command line:

- # Set global level variables, not effective in the current session, effective in the new session

```
set global ob_query_timeout=10000000;
```

- # Set session-level variables, which are effective only in the current session but not in other sessions

```
set session ob_query_timeout=10000000;
```

Introduction to Basic Cluster Usage

4. Set cluster/tenant parameters (or variables)

4.3. Parameters vs variables

Comparison Items	Configuration Items	System Variables
Query Method	You can use the **SHOW PARAMETERS** statement to query. Example: SHOW PARAMETERS LIKE 'schema_history_expire_time';	You can use the **SHOW [GLOBAL] VARIABLES** statement to query. Example: • MySQL Mode SHOW VARIABLES LIKE 'ob_query_timeout'; SHOW **GLOBAL** VARIABLES LIKE 'ob_query_timeout';
Durability	Persisted to internal tables and configuration files, you can query this configuration item in the /home/admin/oceanbase/etc/observer.config.bin and /home/admin/oceanbase/etc/observer.config.bin.history files	Only global-level variables are persisted, session-level variables are not persisted
Life Cycle	The life cycle is **long**, from process startup to exit	The life cycle is **short** and takes effect only after the tenant's Schema is successfully created

Key Points

- OBD & OCP Deploy Tool
- The difference between OBD & OCP
- In the production environment, OceanBase supports the deployment of five centers and five copies in three locations
- FREE! Go! Try it in your environment NOW

OCEANBASE

FROM INTRODUCTION TO PRACTICE

Thank You!

OceanBase Official website:
https://oceanbase.github.io/

GitHub Discussions:
https://github.com/oceanbase/oceanbase/discussions