RGW MULTI-TENANCY

New in version Jewel.

The multi-tenancy feature allows to use buckets and users of the same name simultaneously by segregating them under so-called `tenants`. This may be useful, for instance, to permit users of Swift API to create buckets with easily conflicting names such as “test” or “trove”.

From the Jewel release onward, each user and bucket lies under a tenant. For compatibility, a “legacy” tenant with an empty name is provided. Whenever a bucket is referred without an explicit tenant, an implicit tenant is used, taken from the user performing the operation. Since the pre-existing users are under the legacy tenant, they continue to create and access buckets as before. The layout of objects in RADOS is extended in a compatible way, ensuring a smooth upgrade to Jewel.

ADMINISTERING USERS WITH EXPLICIT TENANTS

Tenants as such do not have any operations on them. They appear and disappear as needed, when users are administered. In order to create, modify, and remove users with explicit tenants, either an additional option –tenant is supplied, or a syntax “<tenant>$<user>” is used in the parameters of the radosgw-admin command.

EXAMPLES

Create a user testx$tester to be accessed with S3:

```
# radosgw-admin --tenant testx --uid tester --display-name "Test User" --access_key TESTER --
```

Create a user testx$tester to be accessed with Swift:

```
# radosgw-admin --tenant testx --uid tester --display-name "Test User" --subuser tester:test
# radosgw-admin --subuser 'testx$tester:test' --key-type swift --secret test123
```

Note: The subuser with explicit tenant has to be quoted in the shell.

Tenant names may contain only alphanumeric characters and underscores.

ACCESSING BUCKETS WITH EXPLICIT TENANTS

When a client application accesses buckets, it always operates with credentials of a particular user. As mentioned above, every user belongs to a tenant. Therefore, every operation has an implicit tenant in its context, to be used if no tenant is specified explicitly. Thus a complete compatibility is maintained with previous releases, as long as the referred buckets and referring user belong to the same tenant. In other words, anything unusual occurs when accessing another tenant’s buckets *only*.

Extensions employed to specify an explicit tenant differ according to the protocol and authentication system used.

S3

In case of S3, a colon character is used to separate tenant and bucket. Thus a sample URL would be:

```
https://ep.host.dom/tenant:bucket
```

Here’s a simple Python sample:

```python
from boto.s3.connection import S3Connection, OrdinaryCallingFormat
c = S3Connection(
        aws_access_key_id="TESTER",
        aws_secret_access_key="test123",
        host="ep.host.dom",
        calling_format = OrdinaryCallingFormat())
bucket = c.get_bucket("test5b:testbucket")
```

Note that it's not possible to supply an explicit tenant using a hostname. Hostnames cannot contain colons, or any other separators that are not already valid in bucket names. Using a period creates an ambiguous syntax. Therefore, the bucket-in-URL-path format has to be used.

SWIFT WITH BUILT-IN AUTHENTICATOR

TBD – not in test_multen.py yet

SWIFT WITH KEYSTONE

TBD – don't forget to explain the function of

rgw keystone implicit tenants = true in commit e9259486decab52a362443d3fd3dec33b0ec654f

NOTES AND KNOWN ISSUES

Just to be clear, it is not possible to create buckets in other tenants at present. The owner of newly created bucket is extracted from authentication information.

This document needs examples of administration of Keystone users. The keystone.rst may need to be updated.