Cloud Computing

Openstack- Private Cloud Computing

Presented by
Prabin Kumar Datta
4th Year 8th Semester, IT, GCECT

Topics

1. Introduction to Cloud Computing
2. Introduction to OpenStack
3. Installation and Configuration- OpenStack

- Introduction to Cloud Computing

Cloud Computing

figure: Cloud computing conceptual diagram

Cloud Computing is Location-independent computing.

Cloud Computing

What is Cloud Computing??

Cloud computing is a computing model, where resources such as computing power, storage, network and software are abstracted and provided as services on the internet in a remotely accessible fashion. Billing models for these services are generally similar to the ones adopted for public utilities.

Some of the key attributes of cloud computing are:

- On-demand availability,
- ease of provisioning,
- dynamic and virtually infinite scalability

Cloud

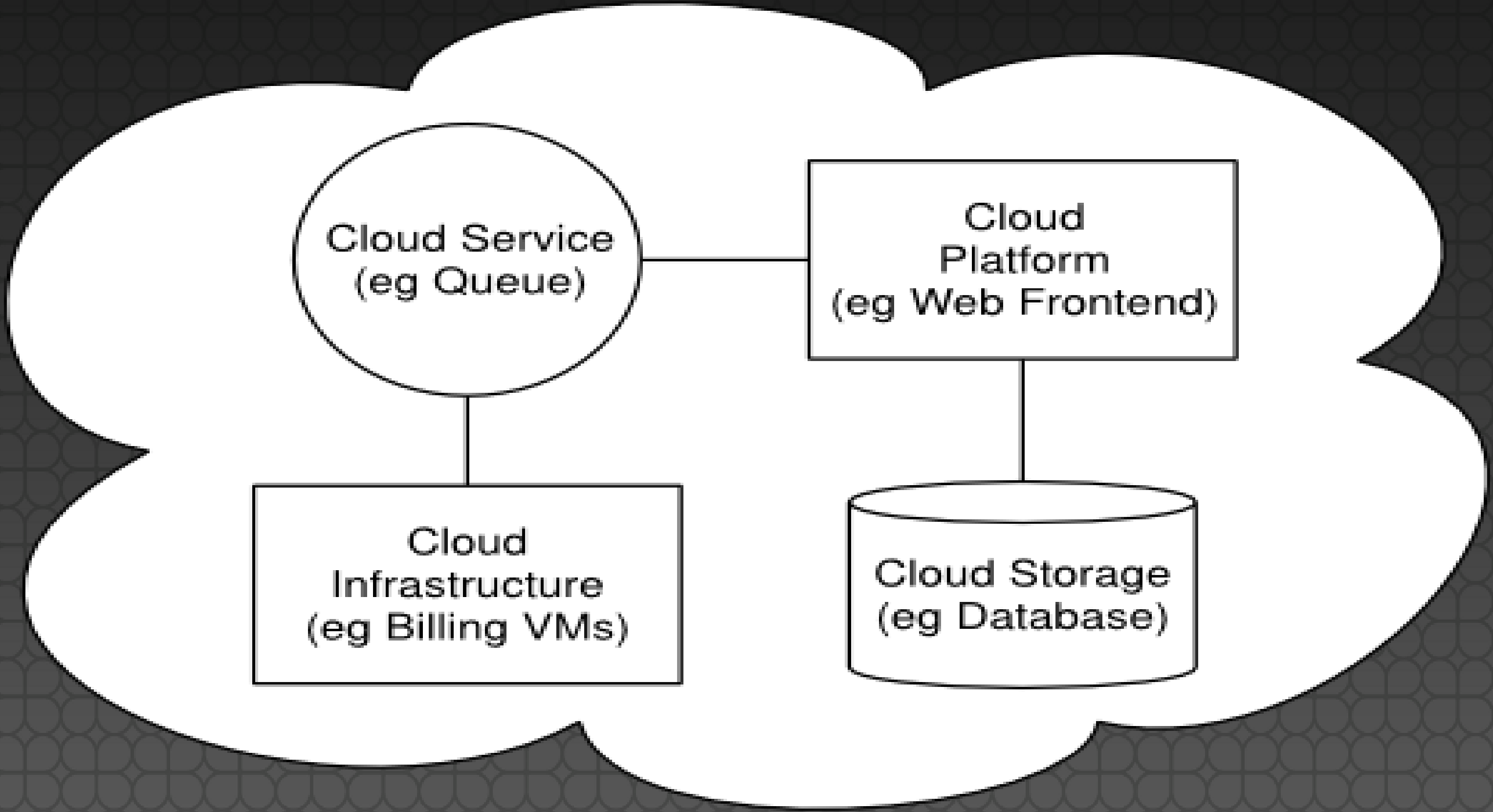
An infrastructure setup using the cloud computing model is generally referred to as the “cloud”.

The following are the broad categories of services available on the cloud:

- Infrastructure As A ***Service*** (IAAS)
- Platform As A ***Service*** (PAAS)
- Software As A ***Service*** (SAAS)

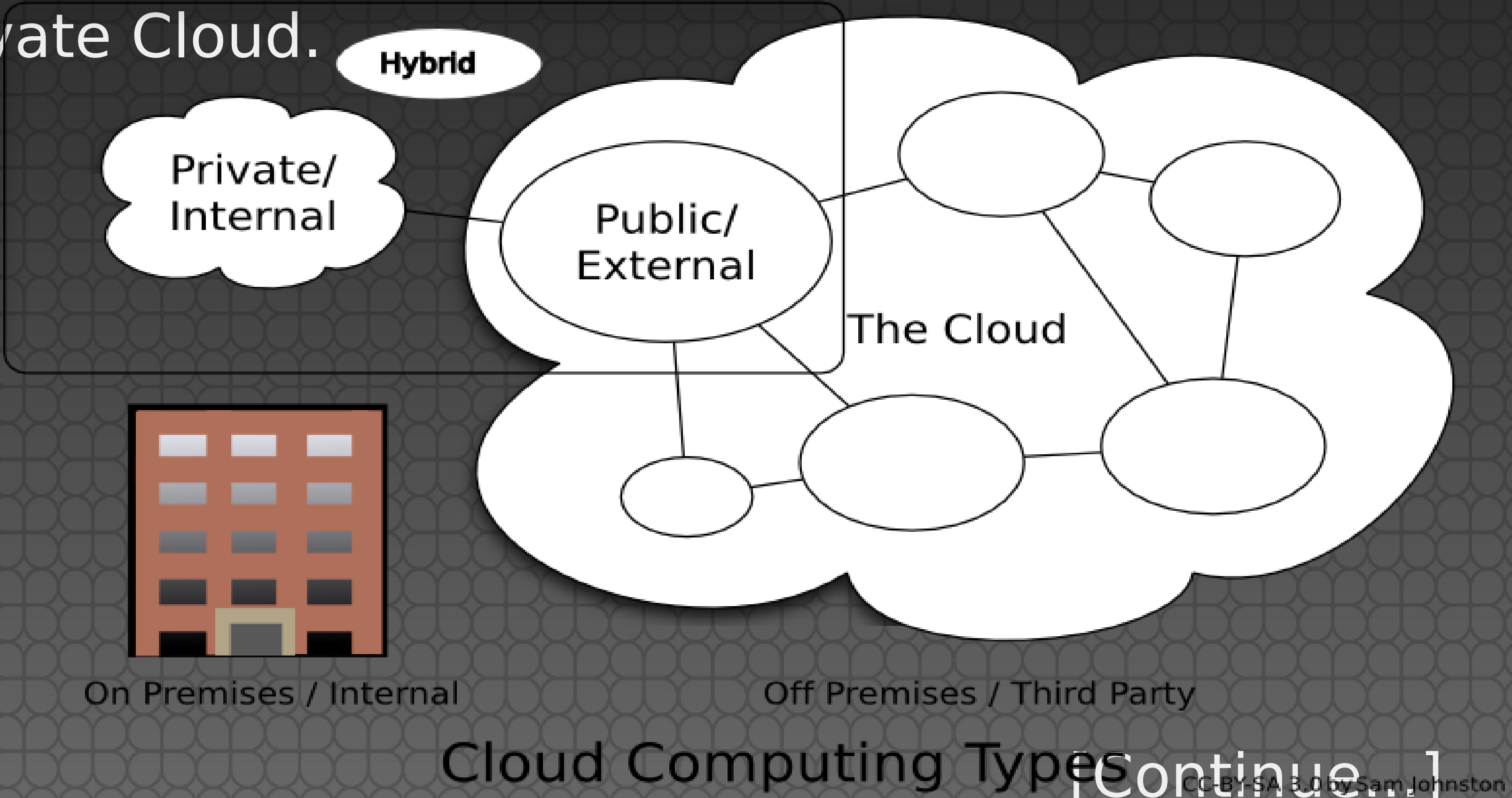
Cloud is generally available as a service to anyone on the internet.

Cloud Computing Architecture



Cloud Computing Types

- Public Cloud.
- Hybrid Cloud.
- Private Cloud.



[Continue...]

Cloud Computing Types

- Public Cloud (also called external cloud):

Public cloud or external cloud describes cloud computing in the traditional main stream sense, whereby resources are dynamically provisioned on a fine-grained, self-service basis over the Internet, via web applications/web services, from an off-site third-party provider who bills on a fine-grained utility computing basis.

- Hybrid Cloud (also called mixed cloud):

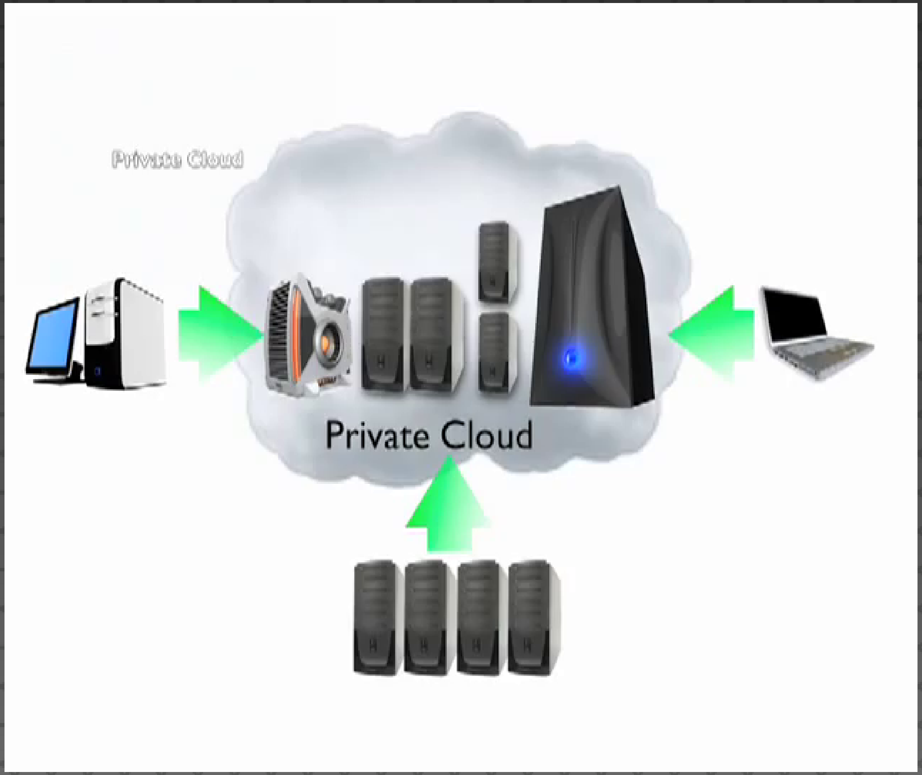
The term Hybrid Cloud has used to mean two separate clouds joined together (public and private).

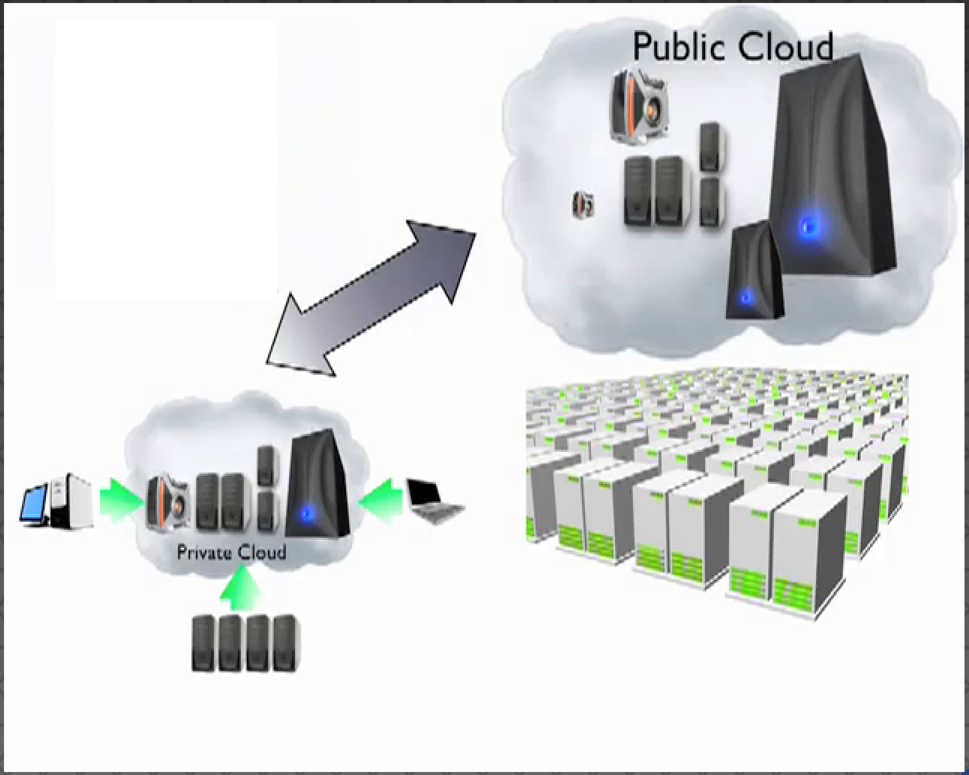
[Continue...]

Cloud Computing Types

- Private Cloud (also called internal cloud)

The computing architecture that provides hosted services to a limited number of people behind a firewall.





fedora

Benefits

3 key benefits Of Cloud Computing are:

- Speed and Time to market.
- Free up your IT staff to do more valuable work.
- Lower your costs

Introduction to OpenStack

OpenStack-swift

OpenStack is a cloud computing framework for building infrastructure as a service (IAAS), and Swift is a subproject which provides a scalable distributed object store.

IAAS:

Infrastructure as a Service is a provision model in which an organization outsources the equipment used to support operations, including storage, hardware, servers and networking components. The service provider owns the equipment and is responsible for housing, running and maintaining it. The client typically pays on a per-use basis.

Architecture

Architecture Overview:

- Proxy Server
- Object Server
- Container Server
- Account Server
- Auth Server

In practice, OpenStack runs Object, Container and Account Server together and breaks out the proxy and auth server

Architecture Overview

- Proxy Server

The Proxy Server is responsible for tying together the rest of the architecture. For each request, it will look up the location of the account, container, or object in the ring and route the request accordingly.

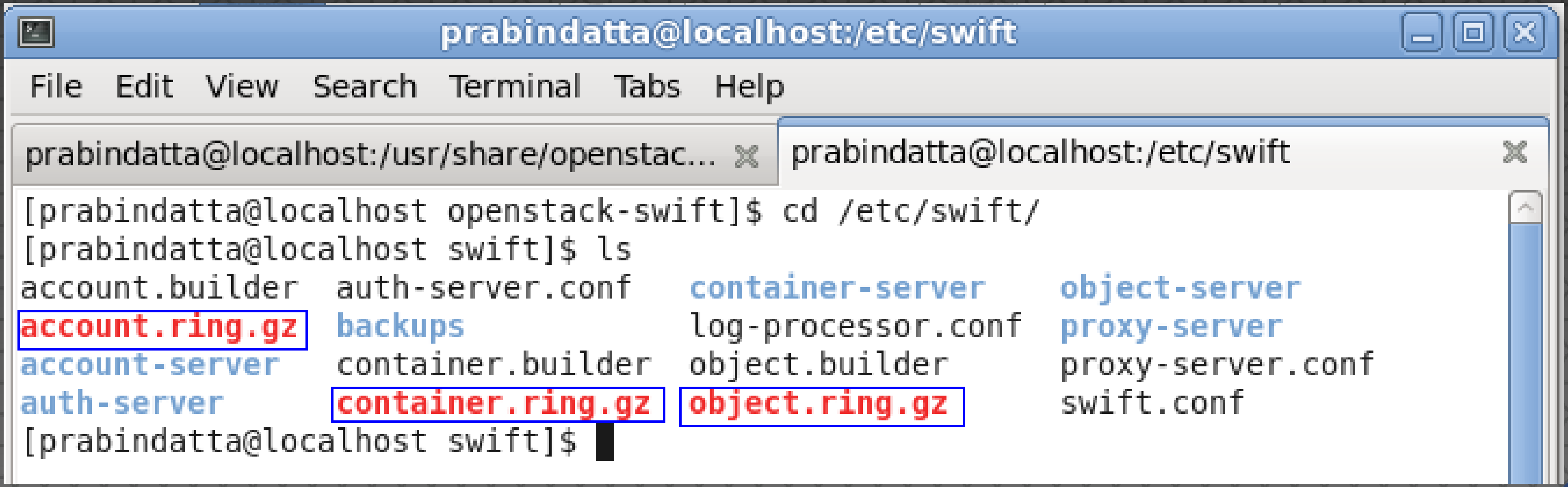
A large number of failures are also handled in the Proxy Server. For example, if a server is unavailable for an object PUT, it will ask the ring for a handoff server and route there instead.

Architecture Overview

What is Ring?

A ring represents a mapping between the names of entities stored on disk and their physical location.

There are separate rings for accounts, containers, and objects.


prabindatta@localhost:/etc/swift
File Edit View Search Terminal Tabs Help
prabindatta@localhost:/usr/share/openstac...
prabindatta@localhost:/etc/swift

```
[prabindatta@localhost openstack-swift]$ cd /etc/swift/
[prabindatta@localhost swift]$ ls
account.builder  auth-server.conf   container-server    object-server
account.ring.gz  backups            log-processor.conf  proxy-server
account-server   container.builder  object.builder      proxy-server.conf
auth-server      container.ring.gz  object.ring.gz      swift.conf
[prabindatta@localhost swift]$
```

fedora

Architecture Overview

- Object Server

The Object Server is a very simple blob storage server that can store, retrieve and delete objects stored on local devices.

Objects are stored as binary files on the filesystem with metadata stored in the file’s extended attributes (xattrs). This requires that the underlying filesystem choice for object servers support xattrs on files.

ext3, have xattrs turned off by default.

Architecture Overview

- Container Server

The Container Server’s primary job is to handle listings of objects. It doesn’t know where those object’s are, just what objects are in a specific container.

- Account Server

The Account Server is very similar to the Container Server, excepting that it is responsible for listings of containers rather than objects.

Installation and Configuration

Required packages

- Openstack-swift
- Openstack-swift-account
- Openstack-swift-auth
- Openstack-swift-container
- Openstack-swift-object
- Openstack-swift-proxy

To Install all packages:

```
# yum install openstack-*
```

Maintaining Log

Package required:

- Syslog-ng

Syslog-ng is used by swift system for log creation, log uploading and log processing.

Configuration file of syslog-ng:

/etc/syslog-ng/syslog-ng.conf

Note: After Configuration restart syslog-ng

[continue...]

Maintaining Log

- Create the log directories:

```
mkdir /var/log/swift/hourly
mkdir /var/log/swift/stats
chown -R <username>:<groupname> /var/log/swift
```

- Create /etc/swift/log-processor.conf
- Add the following under [app:proxy-server] in /etc/swift/proxy-server.conf

```
log_facility = LOG_LOCAL1
```

[continue...]

Maintaining Log

- Create a cron job to run once per hour to create the stats logs. In /etc/cron.d/swift-stats-log-creator
- Create a cron job to run once per hour to upload the stats logs. In /etc/cron.d/swift-stats-log-uploader
- Create a cron job to run once per hour to upload the access logs. In /etc/cron.d/swift-access-log-uploader
- Create a cron job to run once per hour to process the logs. In /etc/cron.d/swift-stats-processor

Large Object Support

Swift has a limit on the size of a single uploaded object; by default this is 5GB. However, the download size of a single object is virtually unlimited with the concept of segmentation.

Segments of the larger object are uploaded and a special manifest file is created that, when downloaded, sends all the segments concatenated as a single object. This also offers much greater upload speed with the possibility of parallel uploads of the segments.

[continue...]

Large Object Support

Segment Objects:

Swift Tool st is used to split a large file into segments and also begin uploading those segments in parallel.

For example:

```
st upload test_container -S 1073741824 large_file
```

This would split the large_file into 1G segments and begin uploading those segments in parallel. Once all the segments have been uploaded, st will then create the manifest file so the segments can be downloaded as one.

Follow st command would download the entire large object:

```
st download test_container large_file
```

Managing the Rings

We need to build the storage rings on the proxy server node, and distribute them to all the servers in the cluster. Storage rings contain information about all the Swift storage partitions and how they are distributed between the different nodes and disks.

- We can build a ring using:

swift-ring-builder <builder_file> create <part_power> <replicas> <min_part_hours>

- Devices can be added to the ring with:

swift-ring-builder <builder_file> add z<zone> <ip>:<port>/<device_name>_<meta> <weight>

[continue...]

Managing the Rings

- Once all of the devices are added to the ring, must run:

  swift-ring-builder <builder_file> rebalance

- See what devices for a server are in the ring:

  swift-ring-builder <builder-file> search <ip_address>

- Removing a device from the ring:

  swift-ring-builder <builder-file> remove <ip_address>/<device_name>

- Removing a server from the ring:

  swift-ring-builder <builder-file> remove <ip_address>

Questions?

Contact:
prabindatta@fedoraproject.org