# CLOUD-OF-CLOUDS STORAGE SERVICE



## Introduction

One of the research directions of the TClouds project is to show how to combine several clouds to build trusted cloud services using resources from a diverse set of commodity public clouds, implementing a cloud-of-clouds able to tolerate faults and intrusions on up to a certain number of providers. The rationale of the approach is that by exploiting replication in diverse and non-correlated infrastructures such as the ones provided by different commodity cloud providers one can avoid any Internet-scale single point of failure.

There are several challenges that need to be addressed to make this approach work. More precisely, it requires replication mechanisms that (1) can tolerate a large spectrum of faults and security threats, (2) can cope with Internet communication unpredictability to effective make multi-cloud systems run smoothly, (3) are compatible with most public clouds available without changing their infrastructures or require their cooperation, and (4) minimize the replication resource overhead to make the solution economically viable for practical applications. These challenges cannot be completely addressed by state of the art replication protocols and thus new techniques must be devised.

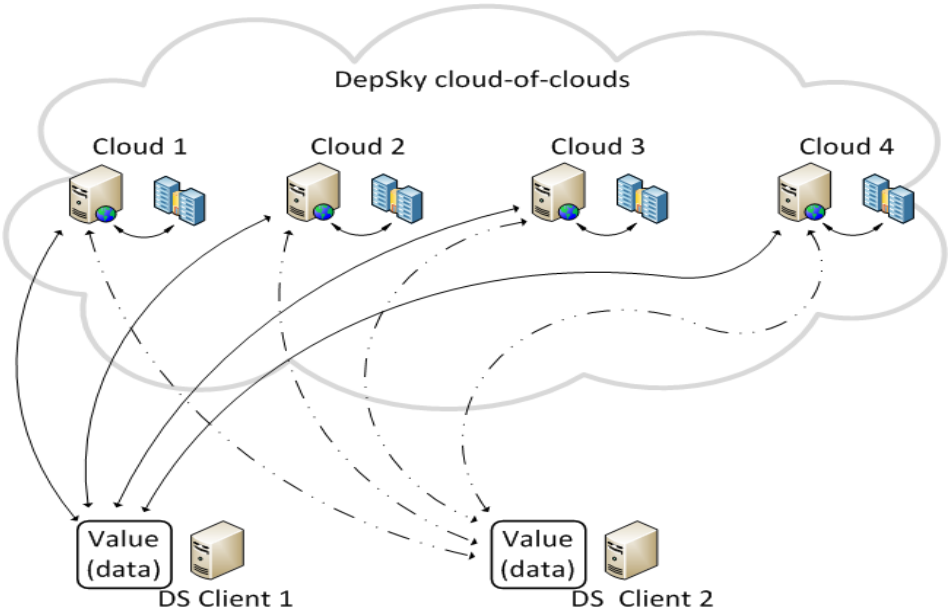


*Figure 1: Cloud-of-Clouds storage architecture*

The *Depsky cloud-of-clouds object storage service* uses object storage services from diverse cloud providers (e.g., Amazon S3, Rackspace Files) to build a dependable object storage service. The figure above illustrates the basic idea behind DepSky.

## Key Techniques

The core of the solution is a set of read/write protocols based on the use of Byzantine quorum replication [2] requiring *3f+1* clouds to tolerate up to *f* unavailable/compromised clouds. This protocol addresses the mentioned requirements in the following way:

(1) The system tolerates arbitrary (a.k.a. Byzantine) faults in order to cope with all possible behavior of a fraction of providers;

(2) The replication protocol operates on an unreliable network in which messages can be lost and delayed and do not require participation of the full set of employed clouds, but only of a subset of them (a quorum [1]), on any step of the protocol execution;

(3) The protocols are completely client-based, in the sense that no specific code is required in the cloud. DepSky assume the clouds provide storage service with standard (RESTful) operations for managing objects and containers (put, get, list, etc.). Moreover, the set of storage clouds do not interact among themselves, but only with the clients;

(4) The storage protocol provided by DepSky employs RAID-like techniques such as *erasure codes* [2] to store only a fraction of the data on each cloud, making the solution cheaper than what would be expected for a full replication scenario (cost less than *N* times as much as single cloud storage, where *N* is the

number of clouds in which the data is replicated).

DepSky employs a *cryptographic secret sharing scheme* [3] to ensure that no single cloud can obtain any information about the stored data. The idea is to make the information stored readable only if the object is read from more than *f* clouds, protecting thus the stored data against leaking to one provider or a minority-collusion of the providers.

The big difference between DepSky and some services that provide secure backup or data sharing is that in the latter, data owners need to completely trust the service provider, while DepSky requires no trust in individual providers.

## Results

For testing DepSky has been deployed in four public clouds, namely, Amazon S3, Rackspace Files, Windows Azure Blob Service and Nirvanix CDN, forming a cloud-of-clouds, that was accessed periodically from several clients around the world for a month. The main results of the experiment were the following [4]. In terms of monetary costs, the use of erasure codes makes DepSky storage cost at most 50% more than single cloud storage (on average). In terms of performance, the DepSky read protocol has lower latency than reading from any single cloud (as data is fetched from the faster cloud) whereas the write protocol latency is close to the time of writing to the slowest cloud.

DepSky is freely available on the Internet as a Java programming library, and its current version can be used to build cloud-of-clouds storage services based on Amazon S3, Google Storage, Rackspace Files and Windows Azure Blob Storage. Moreover, a distributed file system based on DepSky technology is being developed as well.

## References

[1] D. Malkhi and M. Reiter. Byzantine Quorum Systems. Distributed Computing, 11(4), pages 203-213, 1998.

[2] Mi. Rabin. Efficient Dispersal of Information for security, load balancing and fault tolerance. Journal of the ACM, 36(2), pages 335-348. 1989.

[3] A. Shamir. How to share a secret. Communications of the ACM, 22(11), pages 612-613. 1979.

[4] A. Bessani, M. Correia, B. Quaresma, F. André and P. Sousa. DepSky: Dependable and Secure Storage in a Cloud-of-Clouds. In 6th ACM SIGOPS/ EuroSys European Systems Conference (EuroSys'11), pp 31 -45. April 2011.

## Where To Find Depsky?

http://code.google.com/p/depsky/

## Further Information

Further information about Cloud-of-Clouds Storage Service can be found under Deliverable „D2.2.1— Preliminary Architecture of Middleware for Adaptive Resilience".

## Disclaimer

The TClouds project has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement number ICT-257243.

## TClouds at a glance

**Project number:**
257243

**TClouds mission:**

- Develop an advanced cloud infrastructure that delivers computing and storage with a new level of security, privacy, and resilience.
- Change the perceptions of cloud computing by demonstrating the prototype infrastructure in socially significant application areas.

**Project start:**
01.10.2010

**Project duration:**
3 years

**Total costs:**
EUR 10.536.129

**EC contribution:**
EUR 7.500.000

**Consortium:**
14 partners from 7 different countries.

**Project Coordinator:**
Dr. Klaus-Michael Koch
coordination@tclouds-project.eu

**Technical Leader:**
Dr. Christian Cachin
cca@zurich.ibm.com

**Project website:**
www.tclouds-project.eu