PROCEEDINGS OF SCIENCE

~okeanos IaaS

Evangelos Koukis[1]
GRNET
56 Mesogeion Ave, Athens, Greece
E-mail: vkoukis@grnet.gr

Panos Louridas
GRNET
56 Mesogeion Ave, Athens, Greece
E-mail: louridas@grnet.gr

Abstract

This paper introduces ~okeanos, an Iaas platform aiming to deliver advanced computing and storage services to the Greek research and academic community. ~okeanos builds on diverse opensource technologies (Linux/KVM, Google Ganeti, RabbitMQ, Python/Django, Ceph/RADOS) and combines them with custom orchestration software to enable quick, easy and secure access to virtualized resources. Users may build and manage their own isolated, virtual infrastructure inside GRNET's datacenters, using ~okeanos-provided components over a simple, elegant Web UI: Virtual Machines, Virtual Networks (public IPv4/IPv6 and isolated private Ethernets), and Virtual Disks (which may be attached/detached and cloned from existing Images). The paper focuses on the rationale behind ~okeanos, presents current and upcoming features, and discusses its key architectural decisions.

EGI Community Forum 2012 / EMI Second Technical Conference,
Munich, Germany
26-30 March, 2012

[1] Speaker

PoS(EGICF12-EMITC2)007

1. Introduction

This paper presents the design of ~okeanos [1], an IaaS cloud offering virtualized compute and storage resources. It is developed by GRNET, the Greek Research and Technology Network, to be offered to the Greek Research and Academic community. The software powering ~okeanos [2] is available via opensource licenses. ~okeanos offers to its users access to Virtual Machines, Virtual Ethernets, Virtual Disks, and Virtual Firewalls, over a simple web-based UI. It was conceived for easy and secure access to GRNET's datacenters, focusing on user friendliness and simplicity, while being able to scale up to the thousands (of Virtual Machines, users, terabytes of storage).

2. Service description

The goal of the ~okeanos project is to deliver production-quality IaaS to GRNET's direct and indirect customers, IT departments of connected institutions and students/researchers respectively. GRNET operates a working alpha version since July 2011; the alpha offering as of August 2012 comprises ~1300 VMs and ~940 users. The operation of the VMs and the software they run is the responsibility of their owners, as with all Infrastructure-as-a-Service clouds. Based on user feedback the VMs are used for a variety of purposes, including network-facing services (Web servers, FTP and file servers, code repositories), long-running computational workloads (HPC workloads, parallel computing with MPI), deployment of experimental distributed applications for research in distributed systems, and virtual lab environments for education.

The ~okeanos service is a jigsaw puzzle of many pieces:

- Compute/Network Service (codename: cyclades)
- File Storage Service (codename: pithos+)
- Identity Management (codename: astakos)
- Image Registry (codename: plankton)
- Billing Service (codename: aquarium)
- Volume Storage Service (codename: archipelago)

which are combined with a number of activities (monitoring, issue handling, helpdesk operations) to deliver the end-user experience. It goes beyond commercial IaaS providers in several ways: Amazon EC2, and comparable commercial offerings, are not an end-user service, while ~okeanos is designed to be used by people with little computer experience. At the same time it aims to meet the needs of advanced users in technical departments by offering persistent, long-term servers with custom networking capabilities.

The software underlying ~okeanos, called Synnefo, is custom cloud management software. It encompasses a number of distinct *components*, all sharing a single installation and configuration mechanism, to streamline operations. Very early within the project, we made the

decisions to: *a)* build on existing software whenever possible, *b)* target commodity hardware, *c)* release all of the software underlying ~okeanos as opensource.

3. Design and Implementation

3.1 Cyclades: Compute/Network Service

Cyclades is the Compute/Network part of ~okeanos. Its design combines a Google Ganeti backend for VM cluster management with a Python/Django implementation of the user-visible API at the frontend. We opted to reuse Ganeti as a VM management solution in an effort not to re-invent the wheel; Ganeti is a scalable and proven software infrastructure for managing VMs in production environments, and GRNET already had long experience with it, using it to provide VMs to Network Operation Centers. The ~okeanos team is involved in Ganeti development, contributing patches upstream.

Basing our work on Ganeti, we build on a solid, mature core which undertakes most of the low-level VM management operations, e.g., handling of VM creations, migrations among physical nodes, and handling of node downtimes; the design and implementation of the end-user API is orthogonal to VM handling at the backend. Building on Ganeti gave us a head start, allowing us to focus on creating a custom cloud management layer, accesible over a consistent, clean end-user API implementation, and an intuitive Web-based UI. Overall, the software stack is as follows:

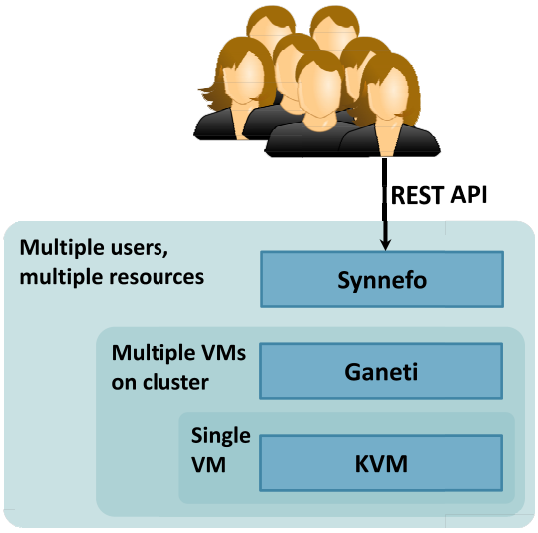

Figure 1: Synnefo software stack

With ~okeanos, users have access to VMs powered by KVM, running Linux and Windows guests on Debian hosts and using Google Ganeti for VM cluster management. The VMs are accessible by the end-user over the Web or programmatically (OpenStack Compute API v1.1). Users have full control over their VMs: they can create new ones, start them, shutdown, reboot, and destroy them. For the configuration of their VMs they can select number of CPUs, size of RAM and system disk, and operating system from pre-defined Images including popular Linux

distros (Fedora, Debian, Ubuntu) and MS-Windows Server 2008 R2. There is an Out-of-Band console over VNC [4] for troubleshooting.

The REST API for VM management, being OpenStack Compute v. 1.1 compatible, can interoperate with 3rd party tools and client libraries. GRNET has added custom extensions for as yet unsupported functionality. It has been implemented in Python, using the Django framework, from scratch.

The ~okeanos UI is written in Javascript/jQuery and runs entirely on the client side for maximum reponsiveness. It is just another API client; all UI operations happen with asynchronous calls over the API.

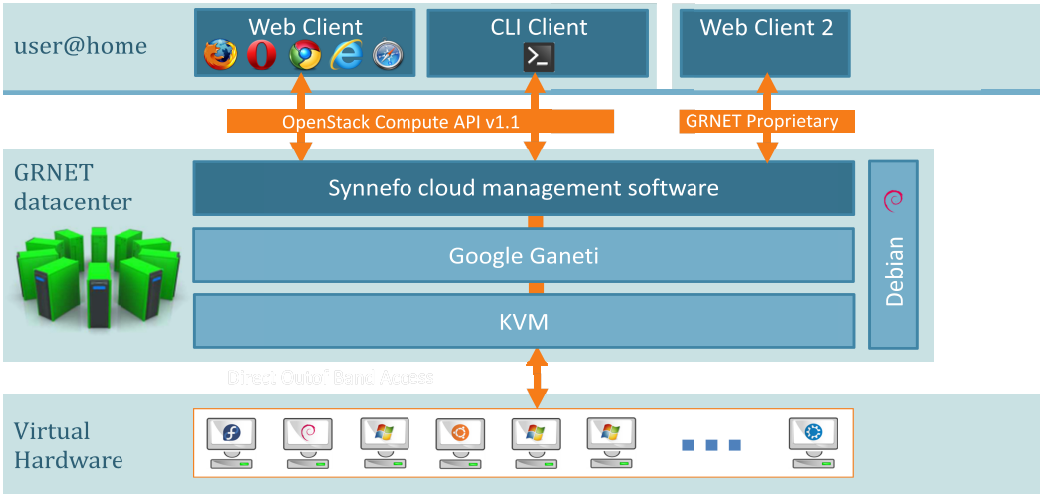

Figure 2: Synnefo platform, from the UI to the backend

The networking functionality includes dual IPv4/IPv6 connectivity for each VM, easy, platform-provided firewalling either through an array of pre-configured firewall profiles, or through a roll-your-own firewall inside the VM. Users may create multiple private, virtual L2 networks, so that they construct arbitrary network topologie, e.g., to deploy VMs in multi-tier configurations. The networking functionality is exported all the way from the backend to the API and the UI.

In the current, alpha deployment, VM disk storage is via redundant storage based on DRDB [3]; VMs survive node downtime, e.g. for planned upgrades, or node failure, facilitating everyday operations tasks while running in production. In the next phases of service deployment, handling of VM storage will be undertaken by Archipelago: an ~okeanos service for handling storage Volumes for VMs as a hierarchy of snapshots and clones.

3.2 Archipelago: Volume Storage Service

Every Volume inside a VM can be thought of as a linearly addressable set of fixed-size *blocks*. The storage of the actual blocks is orthogonal to the task of exposing a single block device for use by each VM. Bridging the gap between the VMs performing random access to Volumes and the storage of actual blocks is Archipelago: a custom storage handling layer which

handled volumes as set of distinct blocks in the backend, a process we call *volume composition.* For the actual storage of blocks we are currently experimenting with RADOS [6], the distributed object store underlying the Ceph parallel filesystem, to solve the problem of reliable, fault-tolerant object storage through replication on multiple storage nodes. Archipelago itself is agnostic to the actual block storage backend.

3.3 Pithos+: File Storage Service

Pithos+ is GRNET's file storage service. It is an implementation of the OpenStack Object Storage API in Python and Django. At the backend, every file is stored as a collection of content-addressable blocks; Using content-based addressing for blocks brings *deduplication* (identical blocks of distinct files are stored only once) and efficient *synchronization*; a client may identify the parts of files which have changed either locally or remotely, and upload or download only the modified parts. Pithos+ comes with a full set of Web-based, command-line and native clients, all making calls to the same API.

Pithos+ is an integral part of ~okeanos: Both system Images and custom, user-provided Images are files on Pithos+ and are registered with Plankton to become available for VM creation. Our goal is for Pithos+ to share the same storage backend with Archipelago, as described in greater detail in Section 4.

3.4 Plankton: Image Registry

Plankton is the Image Registry for ~okeanos. It is implemented as a very thin layer on top of Pithos+; every Image on Plankton is a file on a Pithos+ backend with special metadata. At the frontend, Plankton implements the OpenStack Glance API; at the backend it queries an existing Pithos+ backend. Our current production service runs Plankton and Pithos+ on a single, unified backend: users may synchronize their Images with ~okeanos using the Pithos+ clients, then register them with Plankton, with zero data movement.

3.5 Astakos: Identity Management

Astakos is the identity management service for ~okeanos; it provides the single point of authentication and authorization for the two user-visible ~okeanos services, Cyclades and Pithos+, enabling ~okeanos users to register, login, and handle platform-wide authentication tokens.

Astakos has been designed to support several distinct identity providers at the backend. Currently, it supports local user accounts, Twitter-based authentication, and federated authentication via Shibboleth.

3.6 Aquarium: Billing Service

Aquarium is the common accounting and billing substrate for ~okeanos, currently under development. It receives events from Cyclades, Pithos+, and Astakos, keeping track of resource usage by individual users and billing them in *credits*. In subsequent deployments of the ~okeanos service, Aquarium will be used to enable a policy of fair resource sharing among

users, by assigning every user a number of credits periodically and charging them for their use of distinct resources, e.g., VMs on Cyclades or GBs on Pithos+.

4. Synergy

There is a duality between *Images*, the templates of OSs to be deployed inside VMs and *Volumes*, the virtual disks attached to VMs. Images are static, read-only files, Volumes are dynamic entities, instantiated from Images. *Spawning* a VM is to create a new Volume for it, as a copy of an existing Image. The VM follows its own path, leading its Volume diverging from its initial state. Later, the VM may be *frozen*, meaning a new Image is created as a copy of its Volume at this point in time.

Identifying this duality between Images and Volumes has led to a unified approach for handling storage among all major ~okeanos components: a VM on Cyclades is created from an Image on Plankton, which is a virtual *File* on Pithos+ with extra metadata, and a *Snapshot* on Archipelago. Spawning a VM is *cloning* a Volume from the Snapshot corresponding to this specific Image. Freezing a VM is *making a snapshot* of its corresponding Volume on Archipelago, representing it as a File on Pithos+, and registering it with Plankton. When this process completes, the new Image is available for VM creation on Cyclades.

The following figure shows the synergy between all parts of ~okeanos working together: a unified service from a single API client (kamaki [5]) to the various Synnefo components which implement Astakos, Archipelago, Pithos+, Plankton and Cyclades:

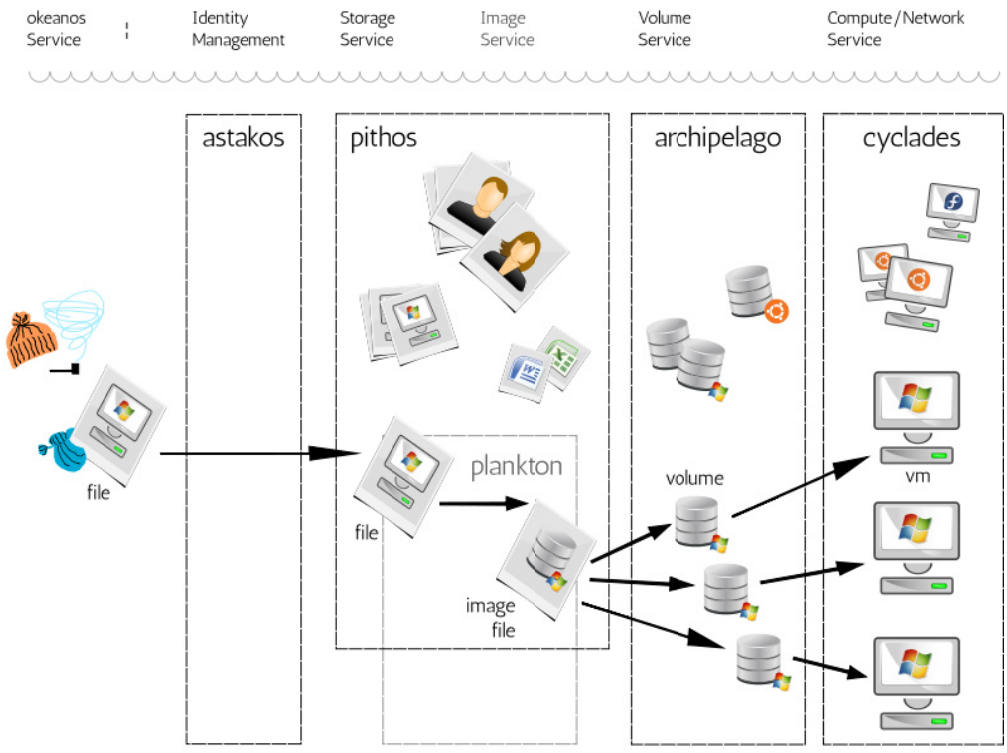

Figure 3: Synergy among ~okeanos components

5. Conclusions - Future Work

We have seen that it is possible to develop in-house a stable, scalable, and user-friendly IaaS, based on open standards. We will offer ~okeanos to the whole Greek research and academic community; all of the underlying software components [2] are made available under free software licenses, 2-clause BSD and the GPL.

Moreover, ~okeanos will be used as the underlying mechanism for a series of other, higher-level services. We are in the process of designing and developing novel PaaS and SaaS, exploiting ~okeanos-provided file handling, Image registration, and lightweight VM creation. We are also exploring the use of ~okeanos for established user groups (like EGI User Communities).

References

[1] https://okeanos.grnet.gr

[2] http://code.grnet.gr/projects/{synnefo, pithos, astakos, aquarium}

[3] http://www.drbd.org

[4] RealVNC - VNC remote access and control technology. http://www.realvnc.com

[5] http://code.grnet.gr/projects/kamaki

[6] Sage A. Weil, Andrew W. Leung, Scott A. Brandt, Carlos Maltzahn. *RADOS: A Fast, Scalable, and Reliable Storage Service for Petabyte-scale Storage Clusters*. Petascale Data Storage Workshop SC07, November, 2007.