Virtualization and Introduction to Public Cloud

René Serral-Gracià[1]

[1]Universitat Politècnica de Catalunya (UPC)

May 9, 2024

Contents

Outline

Objectives

Knowledge

- What is virtualization
- What is the *Cloud*
- What is the *Public Cloud*
- Understand the services present on the cloud
- What *serverless* means

Abilities

- The uses of virtualization
- Manage Virtual Machines
- Be able to determine the best service to a given problem

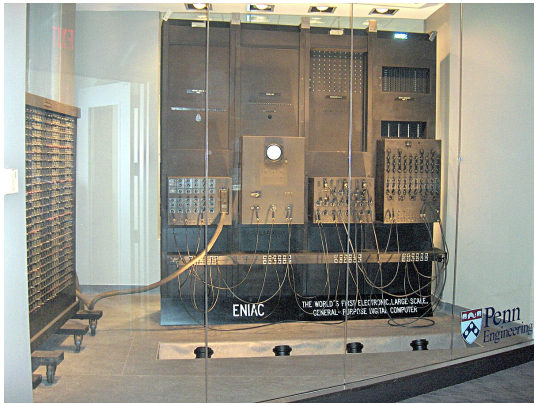
Industrial revolution

- On 1760 We pass from
  - manual labor → steam power
  - rural life → factories, ...

- On 1920 We favor mass production
- huge factories
- gas

3rd Industrial revolution: The digital revolution (1945)

- Semiconductors
- Mainframes
- Computers
- Gaming
- Internet



4rd Industrial revolution: The convergence revolution (2023)

- Cloud Computing
- Data analytics
- Smartcities
- Artificial Intelligence

Outline

What is virtualization...

Virtualization at a glance

An abstraction mechanism to manage
(by partitioning, by merging,...) physical resources

- Virtualization is based on the creation of one (or more) virtual representations of a particular resource
- Examples
  - Local Area Network (VLAN)
  - Web Virtual Hosts
  - Storage Virtualization (e.g., LVM, Cloud storage)
  - Data Virtualization (e.g., seamless access to information)
  - **Hardware Virtualization**

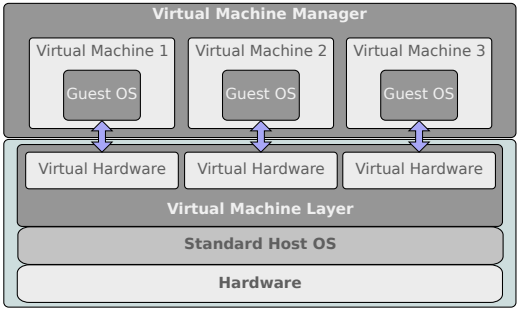
Hardware Virtualization

- The virtualization affects the whole machine where new “virtual” instance is/are created
- The original OS is called *host OS*
- The other “virtual” OS(s) is/are named *guest OS*s
- The new instances work autonomously and use the host OS as proxy with the hardware
- Types of Hardware Virtualization
  - Full Virtualization
  - Paravirtualization
  - Partial virtualization

Outline

Full Virtualization – Overview

- Complete emulation of hardware components
  - Using the legacy OS as proxy to the hardware
- Transparent for guest OS



Full Virtualization – Functionalities and Examples

Functionalities

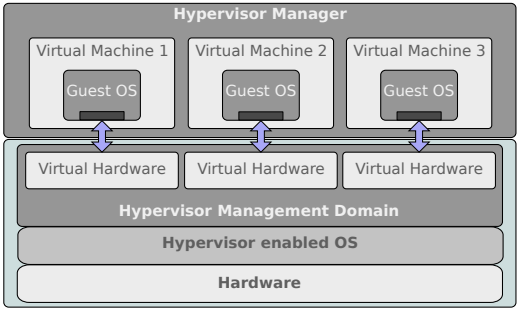
- Memory reservation
- CPU virtualization
- Virtual Network Interfaces

Examples

- Multiplatform
  - VMWare
  - VirtualBox
- Linux: QEmu/KVM
- Windows: Microsoft® Hyper-V Server 2008
- MacOS Parallels

Paravirtualization

- Concurrent execution of different OS: Management Domain controls the rest of OS
- Requires support of the hardware and the OS



Paravirtualization – Functionalities and Examples

Functionalities

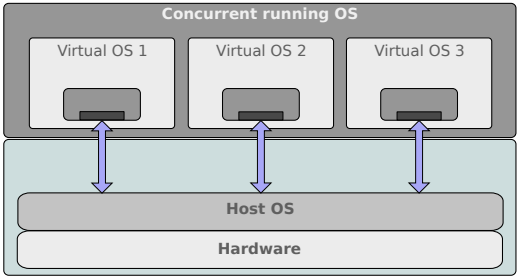
- Memory partitioning
- CPU partitioning
- Network card separation
- Controlled bus access

Examples

- Xen

Partial Virtualization

- Collaboration between host and guests
  - Direct access to the hardware form the guests
  - Can run in userspace
- Requires support of the OS
  - Host and guests use the same OS



Partial virtualization – Functionalities and Examples

Functionalities

- Concurrent execution of various instances of the operating system
- Does not use the virtualization extensions of the hardware
- Based on `namespaces` and `cgroups` (in Linux)

Examples

- OpenVZ
- Solaris Containers
- BSD Jails
- Linux Containers
  - LXC
  - Docker.io

Outline

Main Functionalities

- Machine level backup/restoration
  - Snapshots
  - Regular backups
- Machine Pause/Suspend
- Service Isolation
- Resource limitation (CPU, Memory, I/O, Networking)
- Machine teleporting
  - Memory teleporting
  - Full machine teleporting
- Cloning
- Centralized management

Integrated Management Solutions

- libvirt: `virsh`, `virt-manager` http://www.libvirt.org
- VMWare vSphere http://www.vmware.com/products/vsphere
- OpenNebula: http://www.opennebula.org/
- OpenStack: http://www.openstack.org/

Backup mechanisms

Full Hardware backup

- Use available facilities within the Virtualization Software
- Use a master copy of the system

Backup mechanisms

Full Hardware backup

- Use available facilities within the Virtualization Software
- Use a master copy of the system

Regular backup

- Similar to the legacy case
- Backup shared storage area

Outline

Data center and cloud impact

- Present in our day to day
- It affects our economy
- It forces changes in our business models

What is cloud? NIST says:

What is cloud from NIST

cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction

What cloud really is?

- Outsourcing of resources to a complex system owned by a third party which:
  - Masks the operational details
  - Manages the infrastructure
  - Offers services
  - Has a pay-per-use billing process
- A shift on managing complexity for a company
  - Great CAPEX reduction
  - Fastest deployment
  - Great OPEX increase
  - *Cheaper*... in the short term
- So, it provides
  - Awesome new set of features and available resources
  - Simplified (albeit different) way of managing a system

Goals of adopting the cloud

- We want to reduce:
  - **Time-to-market**
  - **Risk**
  - **Costs**: initial investment vs monthly fee
- Concentrate on the business not the infrastructure
- Do not mind about the company technological resources limitations
- Let experts handle the infrastructure

Characteristics of the cloud

- **Ubiquitous network access**: multiple locations
- **Multitenant**: several concurrent customers with access to the infrastructure
- **Resource Pooling**: resource sharing among customers and dynamically assigned on demand
- **Available resources**: they seem unlimited and adaptable
- **Measured service**: it is controlled and optimized
- **Pay-per-use**

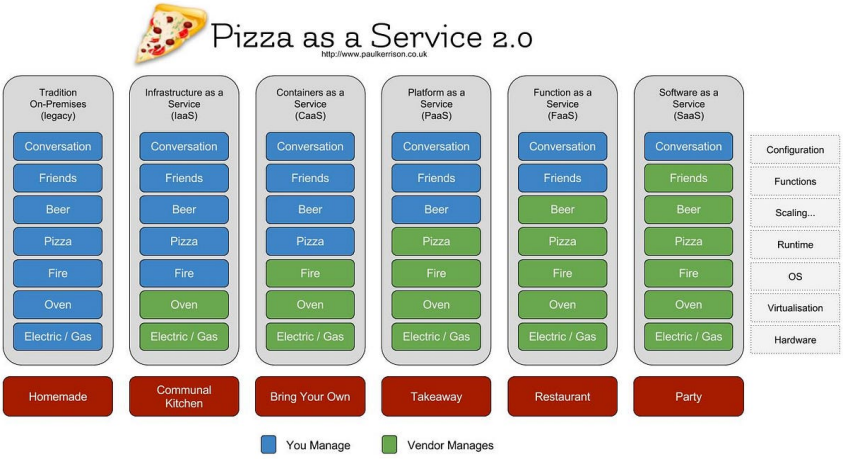
Risks and problems

- What if connectivity drops?
- What if the system fails and all is lost?
- Does regulation allow me to publish this information there?
- How private is my data
- You still need to secure the system
- Lack of control
- Vendor-lock-in

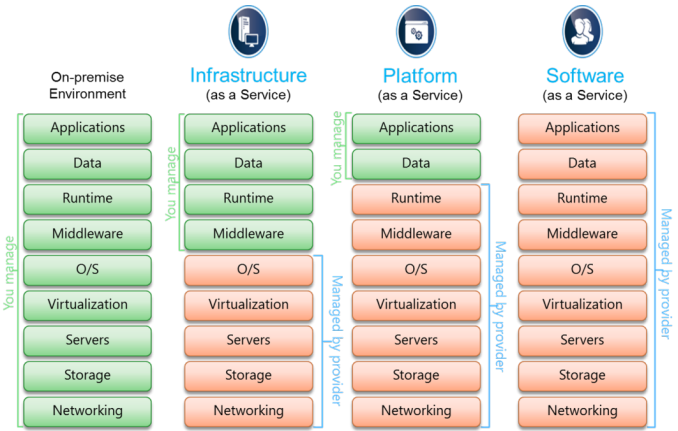
How can we use the cloud?

- We have various models:
  - **Public Cloud**: all infrastructure on the cloud
  - **Private Cloud**: company control of the infrastructure
  - **Community Cloud**: an in-between
- But all deployments may be hybrid, where part of the infrastructure is on a different place

Then the question is: *How much control do we want to keep?*



NIST Service Models



Different providers of Public Cloud

IBM **Cloud**

Software as a Service (SaaS)

- Software licensed on a subscription based fee model
  - Through periodic fees
  - Through advertisement
- Mostly web based
- Examples
  - GMail
  - Facebook
  - Whatsapp

Platform as a Service

- Computing platform
- Customer deploys application using service provider features into the provider's premises
- Provider offers storage, computation, memory, networks, and other necessary facilities
- Preconfigured environment for easy testing and development
- Examples
  - Amazon Web Services (AWS)
  - Heroku

Infrastructure as a Service

- Replacement of the in-house data center
- Provides all necessary infrastructure to work: hypervisor, networking, storage
- The customer is in charge of configuring and administering everything
- Change CAPEX $\rightarrow$ OPEX
- Examples
  - Amazon Web Services (AWS)
  - Rackspace

Outline

AWS Basics

- **Regions**: physical locations of the datacenters
- **Availability Zones** (AZs): different datacenters in the same location (physically separated)
- **Virtual Private Clouds** (VPCs): where resources are interconnected

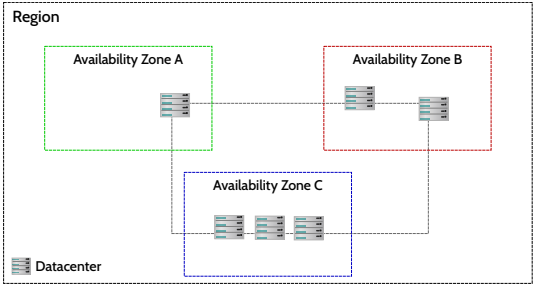
Keys of the cloud

- Highly distributed infrastructure into regions and AZs
- Abstraction of the configuration details through services

AWS Regions

Overall AWS structure

- Each region is built into *Availability Zones* (AZ) - **Datacenters**
- Depending on the *Region* it may have different amount of AZs
- Connected through a low latency high bandwidth redundant links

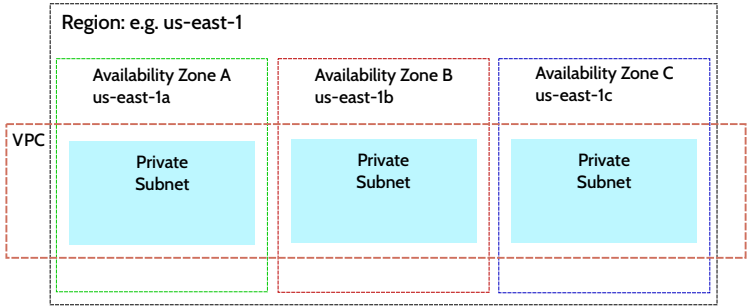


AWS Network characteristics

- All regions are interconnected through the internal Amazon network
- All regions can be directly accessed through the Internet

Overall AWS structure

- A Virtual Private cloud is an abstraction of a subnetwork
- It goes across AZs within the same region
- But a subnet needs to be attached to a single AZ



Amazon Virtual Private Cloud basis

- A VPC is a **virtual network** which allows subnets
- A **subnet** is a range of IP addresses in your VPC within a single AZ. Into a subnet is where you deploy AWS resources
- Subnets require **IP addressing**, which allows to setup IP addresses to AWS resources such as: EC2 instances, NAT gateways, or Network Load Balancers.
- All VPC need **Routing and Route tables** to determine where network traffic from your subnet or gateway is directed to.
- A VPC requires one or more **Gateways**. A gateway connects your VPC to another network

Most relevant services

- AWS IAM
- AWS EC2
- AWS S3 (Serverless)
- AWS Lambda (Serverless)
- Amazon RDS (Serverless)

AWS Identity and Access Management (IAM)

IAM Definition

IAM provides fine-grained permissions to AWS services and resources

- Manages access to resources:
  - **Who?** → Users / Groups of users
  - **can access?** → permissions and privileges
  - **What?** → resources within your organization

- It uses *Attribute Based Access Control*
- It manager per account identities and ties its permissions to them across AWS
- It abstracts permissions in the form of *roles*
- Allows the enforcement of *"Least privilege principle"*

AWS Elastic Cloud Computing (EC2)

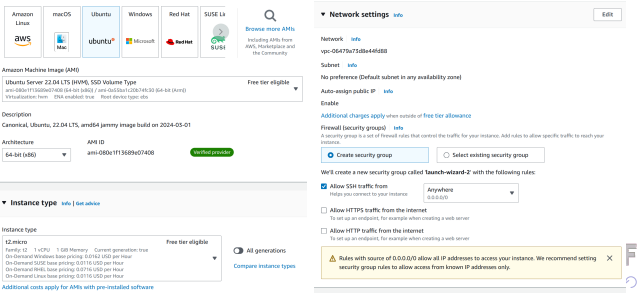
- Provides versatile computing nodes (instances)
- The admin can decide how much memory, disk, CPU, and how many virtual networks an instance will have
- There is a broad selection of preinstalled operating systems
- Amazon handles the infrastructure
- The admin manages:
  - The operating system
  - The applications
  - The configuration
  - The Security Groups
  - The permissions, ...

EC2

EC2 Instances

- **Huge selection of presets**[a]
  - From 0.5GB of memory and 1 vCPU
  - To 128GB of memory and 64 vCPU with GPU acceleration
- **Optimized** for different use-cases
  - General purpose
  - Compute optimized
  - Memory optimized
  - Network optimized
  - Storage optimized
  - HPC Optimized

- Assigned to one or more subnets
- Private or public addresses
- Attached to one or more **security groups** → one per network interface



AWS Security Groups

Security Groups

A security group controls the network traffic that is allowed to reach and leave the resources that it is associated with. When you create a VPC, it comes with a default security group[a]

[a] AWS Documentation

- It must be attached to a VPC → it can only control accesses within that VPC
- They are **stateful**: the response traffic will always be allowed
- They can be applied to virtually anywhere within VPC[1]
- Only users allowed from IAM can modify security groups

[1] There are notable exceptions to this as stated here

What serverless means?

What is a Serverless system?

Serverless computing is a method of providing backend services on an as-used basis. A serverless provider allows users to write and deploy code without the hassle of worrying about the underlying infrastructure. A company that gets backend services from a serverless vendor is charged based on their computation and do not have to reserve and pay for a fixed amount of bandwidth or number of servers, as the service is auto-scaling. Note that despite the name serverless, physical servers are still used but developers do not need to be aware of them[a]

[a] More information here

AWS Simple Storage Service (S3)

Simple Storage Service (S3)

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere[a]

[a] More info here

AWS Simple Storage Service (S3)

- Object oriented storage system
- Each element (object) is composed of a file and metadata about it
- Each object may have a different set of permissions
- The objects are stored into buckets
  - Each bucket allows to control general permissions
  - Who can create, delete, or list objects in the bucket
- You may regard the bucket as a kind of *directory*

Amazon Relational Database Service (RDS)

- Easy to manage relational database
- Allows to use **various engines**: **Aurora MySQL**, **PostgreSQL**, **Oracle**
- **Automatic failover**: the database is automatically replicated to different AZs
- DB infrastructure management with AWS expertise

Managing RDS

The admin just needs to care about the database content and who can access what, the rest is taken care of by AWS staff

AWS Lambda

- Similar to C++ or Python Lambdas
- It provides a serverless (limited) computation node[1]
- It allows the execution of code without provisioning servers
- It is like a funnel $\rightarrow$ requests are sent and responses are delivered

How to use it?

Just upload a ZIP file with the code or container image, the system will allocate the resources and run the code whenever there is an input or a given event

[1] More information in here

AWS example - Architecture