Infrastructure as a Service

Infrastructure as a Service, or IaaS is a service offering by most cloud providers that provides *virtual machines* and the accompanying infrastructure as a service. This lecture will discuss the details of how an IaaS service is built.

Virtual machines

Virtualization is a surprisingly old technology. The first virtualized system was the IBM System/370 mainframe with the VM/370 operating system in 1972. The system was different from how we understand virtualization today, but the goal was the same: separate workloads from each other.

When you think about mainframes you have to consider that these machines were *very* expensive and machine time was a scarce resource. Most programs back in those days were *batch jobs*. They processed a large set of data at once and then terminated.

Initially CPUs in personal computers did not have application separation. The x86 line of Intel CPUs only received the *protected mode* feature with the 80286 in 1982. The operating system (MS/DOS) would run in in *real mode* and applications could then switch into the new mode to isolate applications from each other. One such application making use of the new mode was Windows that ran on top of MS/DOS.

Protected mode introduced the concept of *rings* in the CPU. The operating system *kernel* would run in ring 0, device drivers would run in ring 1 and 2 while applications would run in ring 3. The lower ring number meant the higher privilege level.

> **Note**
>
> Device drivers today typically run on ring 0 instead of 1 or 2.

This ring system allowed the operating system to restrict the higher ring numbers from accessing certain functions or memory locations. However, most applications in the day would violate the restrictions of protected mode and could not run in the new mode.

> **Note**
>
> If you try and set up a really old computer game like Commander Keen in DOSBox you will realize that you have to provide the game itself with very hardware-specific settings. You will, for example, have to provide details for your

> sound card. This is because the game itself incorporated sound card drivers for Sound Blaster 16 or Gravis Ultrasound cards. A game that would do this could not run in protected mode.

To work around the problems with protected mode the 80386 successor introduced virtual mode. The new virtual 8086 mode (VM86) introduced a number of compatibility fixes to enable running old real mode programs in a multitasking environment such as Windows without problems.

For instance the CPU would create a simulated *virtual* memory space the program could write to and translate the virtual addresses to physical addresses internally. It would also capture sensitive instructions and turn them over for control to the kernel.

> **Note**
>
> VM86 does not capture *every* instruction the application runs in virtual mode, only the sensitive CPU instructions. This enables legacy applications to run at a reasonable speed.

In the mid 2000's CPUs became so powerful that it made sense to not only virtualize applications but whole operating systems including their kernel. This allowed multiple operating systems to run in parallel. However, without CPU support only *software virtualization* could be achieved. In other words early virtualization software had to *simulate* a CPU in ring 0 to the guest operating system. Some virtualization techniques, such as Xen required the guest operating system to run a modified kernel to facilitate them running in ring 3. Others employed a number of techniques we won't go into here.

While initially expensive hardware support followed suit. In 2005 Intel added the VT-x (Vanderpool) feature to its new Pentium 4 CPUs followed by AMDs SVM/AMD-V technology in 2006 in the Athlon 64, Athlon 64 X2, and Athlon 64 FX processors.

VT-x and AMD-V added new ring called `-1` to accommodate *hypervisors*. This new ring allowed for separation between several operating systems running at ring 0. Later CPU releases added features such as Direct Input/Output virtualization, network virtualization or even graphics card virtualization. These features allowed for more efficient virtualization and sharing hardware devices between several virtual machines.

Virtualization also gave rise to Infrastructure as a Service. AWS was the first service that offered virtual machines as a service starting in 2006 with a Xen-based offer. They not only offered virtual machines but they did so that a customer could order or cancel the service using an Application Program Interface.

This allowed customers to create virtual machines as they needed it and they were billed for it on an hourly basis. (Later on AWS and other cloud providers moved to a per-second billing.)

The presence of an API makes the difference between IaaS and plain old virtual machines as a service. IaaS allows a customer to scale their application dynamically according to their current demand.

Typical instance types

When the cloud became popular in the late 2000s several providers attempted to offer a service that was fully dynamic in their sizes. The customer could set how many GB of RAM they needed and how many CPU cores. However, this model has been phased out by most providers since it is difficult to manage such a dynamic environment.

Instead cloud providers nowadays opt to offer fixed machine sizes (think of t-shirt sizes). To accommodate high-CPU and high RAM workloads there are several different instance types, typically:

- **Shared CPU:** These are small instances where a single CPU core is shared between multiple virtual machines, sometimes leading to high steal time. Sometimes this offering includes a *burst* capability (such as the Amazon T instances) where a VM can temporarily use more CPU.
- **Standard, dedicated core CPU:** These instance types receive one or more physical cores leading to a more stable performance without the ability to burst beyond their limits.
- **High CPU:** These instance types are usually hosted on physical servers that have a very high CPU to RAM ratio. Accordingly, the virtual machine offering includes more CPU than the standard offering.
- **High RAM:** This offering is the exact opposite of the high CPU offering. The machines on offer here include more RAM with very little CPU.
- **Storage:** These instance types contain large amounts of local storage (see below in the storage section).
- **Hardware-specific:** These instance types offer access to dedicated hardware such as graphics cards (GPUs) or FPGAs.

Automation

As discussed before, that makes an IaaS cloud provider a cloud provider is the fact that they offer an API to automate the provisioning and deprovisioning of virtual machines as needed. However, that's not all. Simply starting a virtual machine is not enough, the software needs to be installed in it.

Initially this problem would be solved by creating *templates* for the operating system that launches. In larger cloud setups these templates included a pre-installed agent for configuration management that would report to a central service and fetch its manifest of software to install.

Thankfully in the last decade a lot has happened and Cloud Init has established itself as a defacto standard in the IaaS world. Every cloud provider nowadays offers the ability to submit a *user data* field when creating a virtual machine. This user data file is read by Cloud Init (or its Windows alternative Cloudbase Init) and is executed at the first start of the virtual machine.

A DevOps engineer can simply inject a script that runs at the first start that takes care of all the installation steps required.

Tools like Terraform or Ansible assist with managing the whole process of provisioning the virtual machines and supplying it with the correct user data script.

Virtual machine pools

One other use of user data are virtual machine pools. Each cloud provider adopts a different name for them, ranging from instance pools to autoscaling groups. The concept is the same everywhere: you supply the cloud with a configuration how you would like your virtual machines to look like and the cloud will take care that the given number of machines are always running. If a machine crashes or fails a health check the cloud deletes the machine and creates a new one.

The number of machines in a pool can, of course, be changed either manually or in some cases automatically using rules for automatic scaling.

Combined with the aforementioned user data this can be a very powerful tool to create a dynamically sized pool of machines and is the prime choice for creating a scalable architecture.

These pools are often integrated with the various load-balancer offerings cloud providers have in their portfolio to direct traffic to the dynamic number of instances. Some cloud providers integrate them with their Functions as a Service offering as well allowing you to run a custom function whenever a machine starts or stops. This can be used to, for example, update your own service discovery database.

Storage

Local storage

Block storage

Network file systems

Object storage

Network

Firewalling

Network load balancers

Private networks

DNS

VPNs, private interconnects, and routing services

Monitoring

Automation