Multitenancy and the Noisy Neighbor Problem

This lesson discusses single tenancy, multi-tenancy, and the noisy neighbor problem.

We'll cover the following

Let's begin the lesson with a quick look at two key terms associated with application hosting:

- Single-tenant
- Multi-tenant

What do these terms really mean? Why are they important in cloud computing?

Let's find out.

Tenancy in application hosting

Single-tenant means the resources of the server are utilized by just one tenant.

What is a tenant?

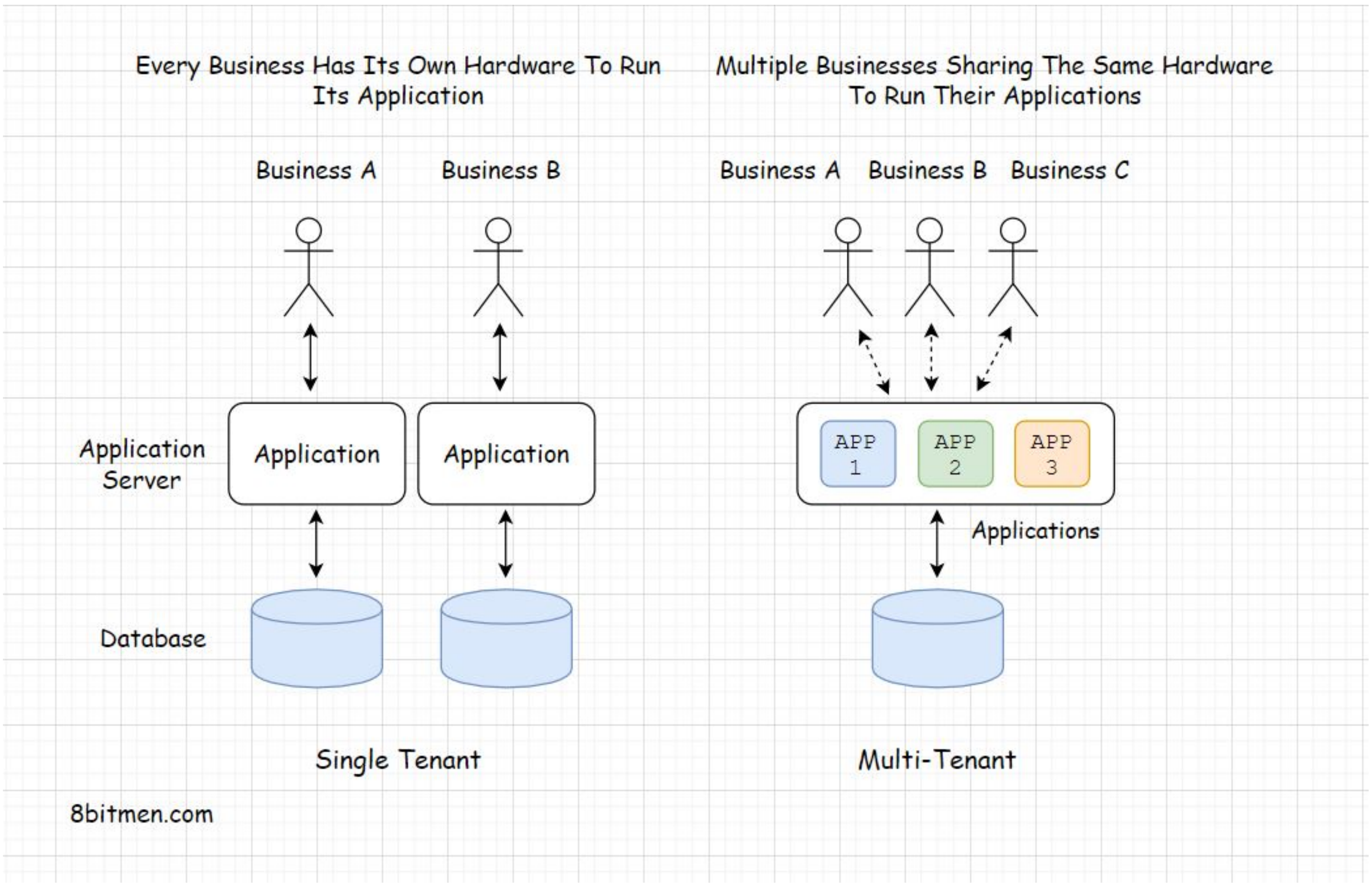
The meaning of the term tenant varies based on the context. From an application hosting standpoint, a tenant means a business that has deployed its workload on the server.

Single-tenant means that just one business entity has deployed its workload on the server as opposed to multiple applications of multiple businesses utilizing the resources of a single server.

When multiple businesses consume the resources of a server, it's called *multi-*

tenancy. In this multitenancy scenario, multiple workloads share the resources of

the server such as *memory, compute power, network bandwidth*, and so on, staying isolated and invisible to each other.



Multitenancy is largely used in cloud computing to share the resources of the infrastructure amongst multiple businesses. This brings down the costs significantly, enabling the provider to achieve economies of scale.

Real-life examples

Shared hosting

Shared hosting of a simple website or a blog is a common example of this. When we run a WordPress blog, or any other simple website, in its initial phase, we deploy our blog on shared hosting to keep the costs low.

As we gain traction and need more resources, we may transition to dedicated hosting. The terms dedicated hosting and isolated hosting can be used interchangeably with single-tenant hosting.

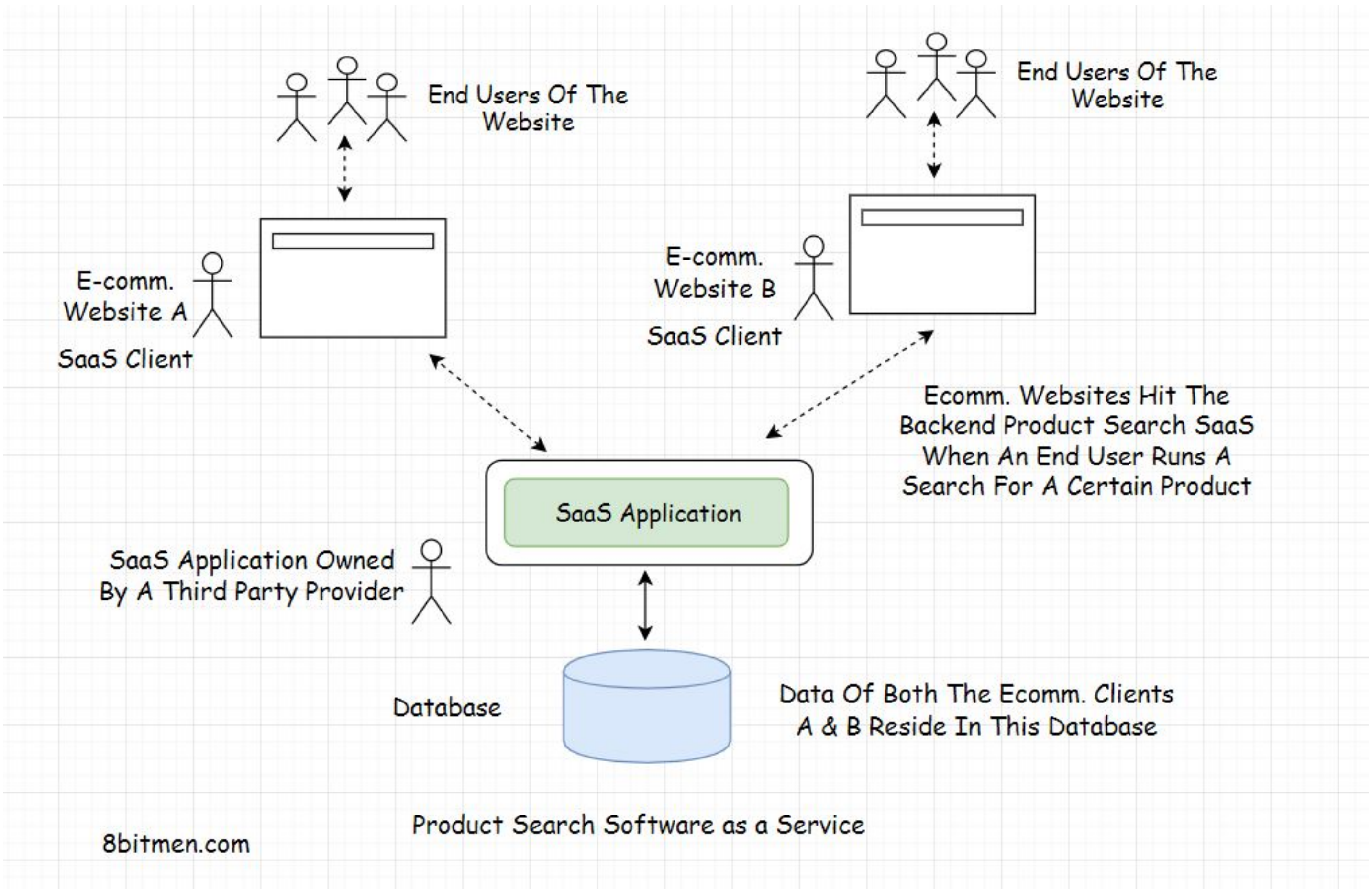
Software as a Service (SaaS)

SaaS is one more example of multi-tenant hosting. In this scenario, the same software is used by multiple businesses. They share the resources of the infrastructure that hosts the software.

Think of a search service offered by a third-party provider that we can plug-in into our e-comm. website to enable the users to search through the inventory of the website.

To make this work, we would have to stream our inventory data to the third-party *SaaS* database and let it host our data for us along with the data of the other businesses. When a user runs a search for a product on the website, the query is routed to the backend of the *SaaS* provider.

In this SaaS use case, multiple businesses use the same database, same storage mechanism, same *OS*, and same hardware to power the search on their website.


End Users Of The Website
End Users Of The Website
E-comm. Website A
SaaS Client
E-comm. Website B
SaaS Client
Ecomm. Websites Hit The Backend Product Search SaaS When An End User Runs A Search For A Certain Product
SaaS Application
SaaS Application Owned By A Third Party Provider
Database
Data Of Both The Ecomm. Clients A & B Reside In This Database
Product Search Software as a Service
8bitmen.com

On the contrary, in single-tenancy, the resources are not shared between businesses. This provides more control and security to a single business. There is more room for customization. With multi-tenancy, there may be security risks involved because the data of multiple businesses reside in the same machine.

The noisy neighbor problem is another ill effect of multi-tenancy.

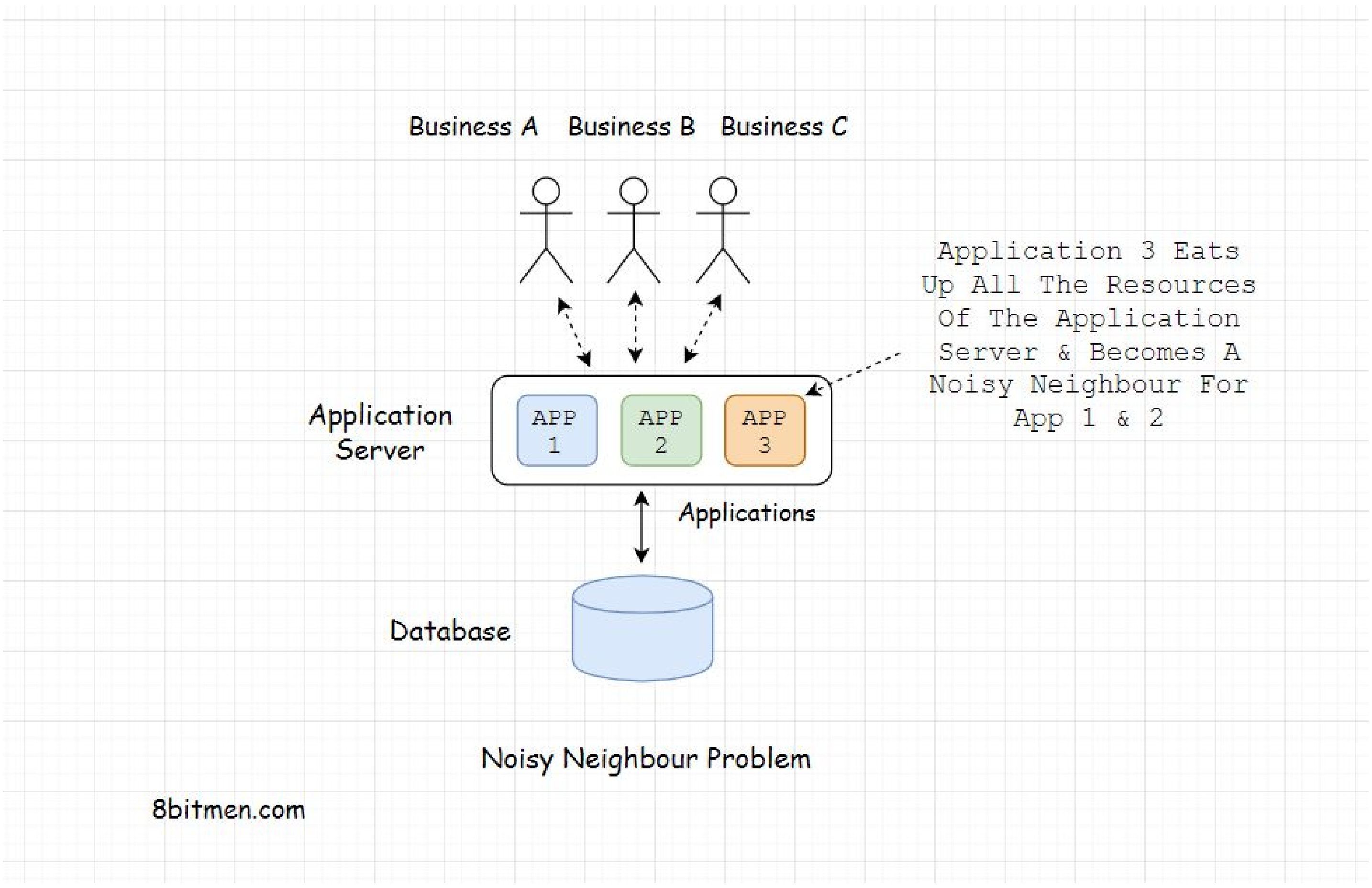
Let's discuss what it is?

The noisy neighbor problem

The noisy neighbor problem

When multiple workloads are deployed on a single machine, one particular or a few workloads among them may eat up all the resources of the machine and leave the others with a resource crisis.

This hogging up of resources by one workload makes the other workloads deployed in the same machine suffer from a performance standpoint. Their throughput takes a hit. This issue in multi-tenancy is known as the *noisy neighbor problem*.



Hosting providers adopt different strategies to deal with this issue. They either overprovision the hardware to avoid it, or they kill the processes of the noisy tenant, subsequently intimating the business. Effective application monitoring is essential for tackling the noisy neighbor issue.

Deploying our application on the bare metal servers, helps us get rid of the noisy neighbor problem completely.

What are *bare metal servers*? We'll find out in the next lesson.