Web services

Part 4

Lecture #4

- Clustering
- Virtualization
- Docker
- Load-balancer
    - Nginx
    - HAProxy
- Project_4

What’s a cluster?

Clustered web hosting

Clustered hosting is a type of web hosting that spreads the load of hosting across multiple physical machines, or node, increasing availability and decreasing the chances of one service (e.g., FTP or email) affecting another (e.g., MySQL).

Many large websites run on clustered hosting solutions, for example, large discussion forums will tend to run using multiple front-end web-servers with multiple back-end database servers.

Clustered web hosting

Typically, most hosting infrastructures are based on the paradigm of using a single physical machine to host multiple hosted services, including web, database, email, FTP and others. A single physical machine is not only a single point of failure, but also has finite capacity for traffic, that in practice can be troublesome for a busy website or for a website that is experiencing transient bursts in traffic.

Clustered web hosting

By clustering services across multiple hardware machines and using load balancing, single points of failure can be eliminated, increasing availability of a website and other web services beyond that of ordinary single server hosting. A single server can require periodic reboots for software upgrades and the like, whereas in a clustered platform you can stagger the restarts such that the service is still available whilst still upgrading all necessary machines in the cluster.

Clustered hosting is similar to cloud hosting, in that the resources of many machines are available for a website to utilize on demand, making scalability a large advantage to a clustered hosting solution.

Clustered web hosting

Advantages:

- Clustering servers is completely a scalable solution. You can add resources to the cluster afterwards.
- If a server in the cluster needs any maintenance, you can do it by stopping it while handing the load over to other servers.
- Among high availability options, clustering takes a special place since it is reliable and easy to configure. In case of a server is having a problem providing the services furthermore, other servers in the cluster can take the load.

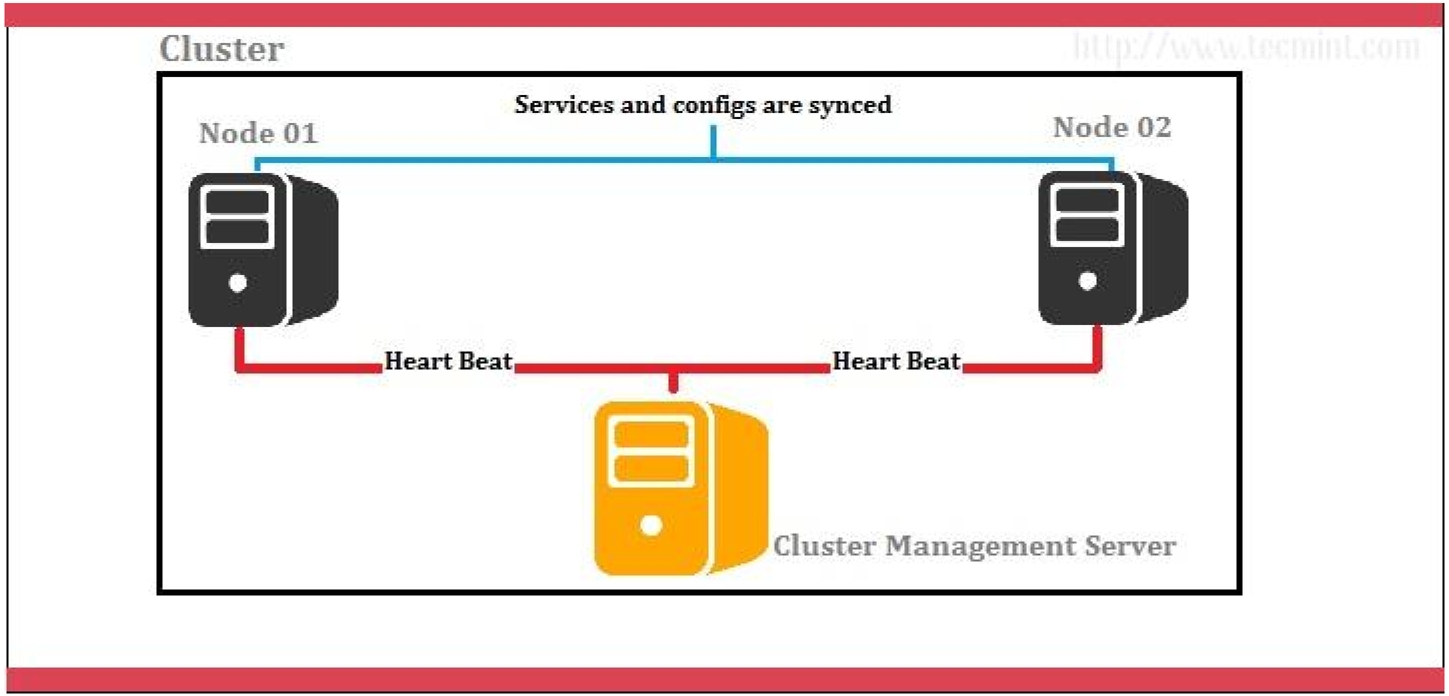
Clustered web hosting

Disadvantages:

- **Cost is high**. Since the cluster needs good hardware and a design, it will be costly comparing to a non-clustered server management design. Being not cost effective is a main disadvantage of this particular design.

- Since clustering needs more servers and hardware to establish one, monitoring and maintenance is hard. Thus increase the infrastructure.

Clustered web hosting



How to build a cluster?

- Physical cluster
  - You have many servers
- Virtual cluster
  - You have less servers but more powerful ones
  - Virtualization
  - Containers

Virtualization

Virtualization

In computing, virtualization refers to the act of creating a virtual (rather than actual) version of something, including virtual computer hardware platforms, operating systems, storage devices, and computer network resources.

You can virtualize:

- A cpu architecture (ARM on x86_64)
- A router
- An Operating System
- A network architecture
- A usb device (virtual printer)

Virtualization

Different types of hardware virtualization include:

- **Full virtualization** – almost complete simulation of the actual hardware to allow software, which typically consists of a guest operating system, to run unmodified.
- **Partial virtualization** – some but not all of the target environment attributes are simulated. As a result, some guest programs may need modifications to run in such virtual environments.
- **Paravirtualization** – a hardware environment is not simulated; however, the guest programs are executed in their own isolated domains, as if they are running on a separate system. Guest programs need to be specifically modified to run in this environment.

Virtualization

Advantages:

- Save energy, go green

  Since you have less you hardware, you consume less power

- QA/lab environments

  You can test a totally new environment without buying new HW

- Faster server provisioning

  Create new VM’s on the fly

- Reduce hardware vendor lock-in

  You can emulate an OS but also hardware

Virtualization

Advantages:

- Increase uptime

  You can “pop” a new instance while you are fixing a broken one

- Improve disaster recovery

  Migrate your VM into another server just by copying few files and not the whole file-system or OS configuration

- Isolate applications

  Application can run sandboxed without interfering with your other running applications

- Extend the life of older applications

  You can run your old application that is running on an old windows 98 while the other ones run on Windows 10

Virtualization

Disadvantages

- High Risk in Physical fault

  If one of your component is broken, ALL your VM’s will suffer from this
- Performance

  Performances are not as good as a direct access to your hardware since your host operating system will have to perform more operations.
- Complicated

  Remember when you tried to use the VirtualBox image on VMWare…
- Cost

  VMWare is not free for example. Some virtualization systems will charge you per core on your server.
- Not Supported by All Applications

  Database or application that requires GPU computing are not (well) designed for this (yet)

Virtualization

Virtualization is not new, it exists since 1964 (IBM CP-40) but it has been really trendy when we started to talk about Green-computing.

It will not solve all your problems and it’s not the best solution all the time. The best would be to use virtualization with hardware redundancy.

You might save time/money by requiring less servers but you will have to buy high-quality components.

You kinda know how virtualization is working and the benefits it can brings you.

However, this solution is quite heavy especially when you have **limited resources**.
This is why we will focus on **Docker**

Docker

Docker

Docker is an **open-source** project that automates the deployment of Linux applications inside software containers.

Docker **containers** wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in.

Docker

Docker, a new container technology, is hotter than hot because it makes it possible to get far more apps running on the same old servers and it also makes it very easy to package and ship programs.

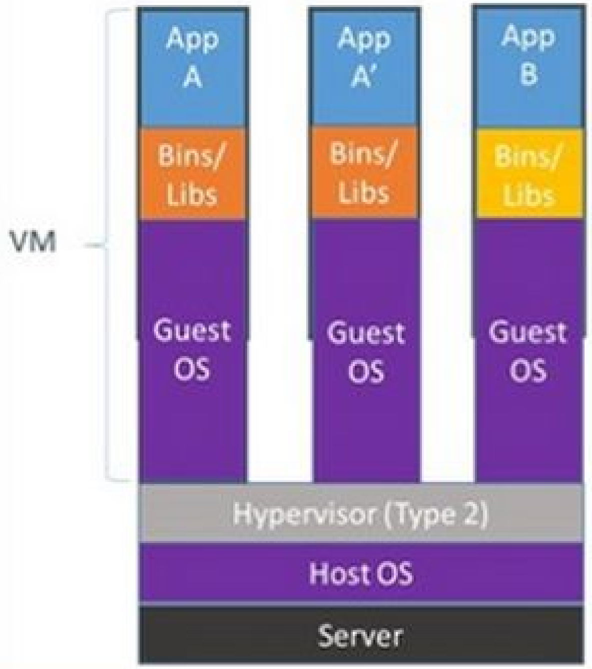
Docker

Containers use shared operating systems. That means they are much more efficient than hypervisors in system resource terms. Instead of virtualizing hardware, containers rest on top of a single Linux instance. This in turn means you can “*leave behind the useless 99.9% VM junk, leaving you with a small, neat capsule containing your application*”

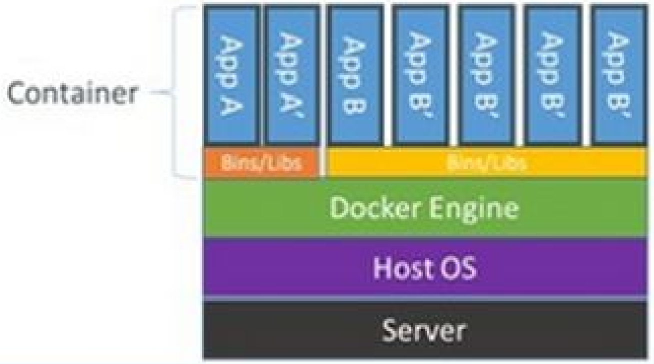
Docker

However, the more you need to run containers on your server, the more you will need CPU cores and RAM.

Containers vs. VMs



Containers are isolated, but share OS and, where appropriate, bins/libraries



Docker

Advantages

- **Portable deployment** of applications as a single object versus process **sandboxing**;
- **Application-centric** versus machine/server-centric;
- Supports for automatic container builds;
- Built-in **version tracking**;
- Reusable components;
- Public registry for **sharing containers**; and
- A growing tools **ecosystem** from the **published API**.

Docker

Disadvantages

- Not right for all tasks
- **Weaker** isolation
- **Limited** tools
- You use **Windows/Unix**
- You can’t emulate any hardware
- Docker mirrors are very slooooooooow in china

How to use docker?

It’s super easy, the documentation will help you a lot :p

Anatomy of a project using docker

You just need to have:

- Your source code
- Dockerfile
- .dockerignore

Dockerfile

Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Using docker build users can create an automated build that executes several command-line instructions in succession.

This file is not really required but it will help you to push your images online faster and help you to deploy containers to production.

Dockerfile

```
FROM debian:jessie
COPY ./bin/api /go/bin/api
COPY ./entry-point.sh /entry-point.sh

RUN echo "Asia/Shanghai" > /etc/timezone
RUN dpkg-reconfigure -f noninteractive tzdata
RUN date

ENTRYPOINT ["/entry-point.sh"]

EXPOSE 8081

CMD ["/go/bin/api"]
```

Load-balancing

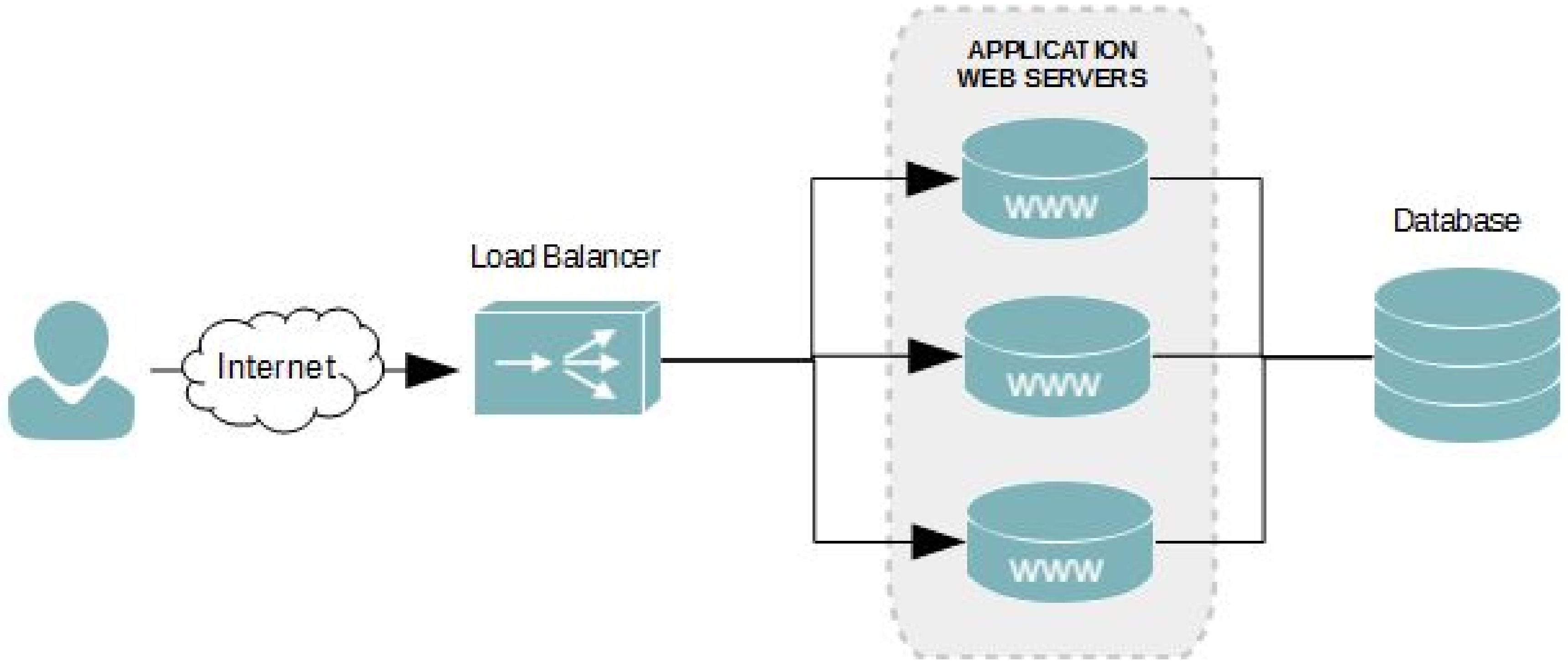
Load-balancing

In computing, **load balancing improves** the distribution of **workloads** across multiple computing resources, such as computers, a computer cluster, network links, central processing units, or disk drives. Load balancing aims to **optimize** resource use, **maximize throughput**, minimize response time, and avoid overload of any single resource.

Using multiple components with load balancing instead of a single component may **increase reliability** and availability through redundancy.

Load balancing usually involves dedicated software or hardware, such as a multilayer switch or a Domain Name System server process.

Load-balancing



Load-balancing methods

- **Round Robin** – Requests are distributed across the group of servers sequentially.
- **Least Connections** – A new request is sent to the server with the fewest current connections to clients. The relative computing capacity of each server is factored into determining which one has the least connections.
- **IP Hash** – The IP address of the client is used to determine which server receives the request.

Load-balancing

Advantages

- **Distributes** client requests or network load efficiently across multiple servers
- **Ensures high availability** and **reliability** by sending requests only to servers that are online
- Provides the **flexibility** to **add** or **subtract** servers as demand dictates

Load-balancing

Disadvantages

- Potential uneven spread of workloads across cluster nodes resulting in slow user response times and high latency for the application.
- Cannot detect service outage. It can only detect server outage by IP address. If a particular server service fails, WNLB cannot detect the failure and will still route requests to that server.
- Unable to consider each server current CPU load and RAM utilisation when distributing client load.
- Only supports source IP address affinity / persistence.

Load-balancer

Load-balancers are mostly software pieces but there are also hardware solutions.

HDL (Hardware Load-balancing Device) minimizes the probability that any particular server will be overwhelmed and optimizes the bandwidth available to each computer or terminal. In addition, the use of an HLD can minimize network downtime, facilitate traffic prioritization, provide end-to-end application monitoring, provide user authentication, and help protect against malicious activity such as denial-of-service (DoS) attacks. They are extremely fast but it will require you a higher budget.

Load-balancer

Since we will have to deploy a cluster with only 2 instances, we can focus on software ones which are easier to configure but still really powerful

I recommend you:

- Nginx

or

- HAproxy

From my point of view, Nginx is really simple to use, and HAProxy is faster.

Nginx configuration

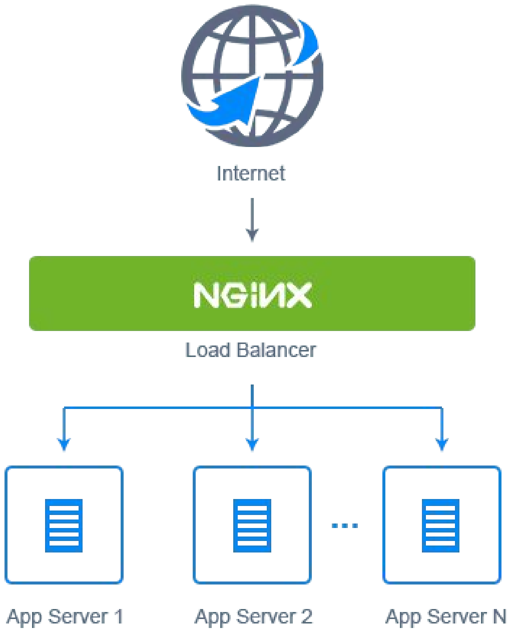
```
http {
    upstream myapp1 {
        server <ip1>:<port>;
        server <ip2>:<port>;
        server <ip3>:<port>;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://myapp1;
        }
    }
}
```

For HAProxy, I suggest you to check it by yourself

Nginx configuration



project_4

project_4

In this project, you will have to build a simple image resizing service and “Dockerize” it.

project_4

You will use docker, so I allow you to use any programming language you want in the following list:

Python 3.x

I know you all love python now,
so no need more programming languages in this list

project_4

You will save you source code on gitea under the name **project_4**

Required files:

- Dockerfile
- .gitignore

Optional file:

- .dockerignore

project_4

Your web-service will have to listen on ***:8888**

When I POST a picture, you will have to return it with the same picture but with a smaller/bigger size like.

POST /resize?scale=2 (this means that the returned picture should be twice bigger)

POST /resize?scale=0.5 (here the picture is twice smaller)

POST /resize?scale=abc (status code 400 without any body)

project_4

Extra requirements

- You need to support *.**jpg**, *.**jpeg**, *.**png**, *.**bmp**, *.**gif**
- If the sent file is not correct, you need to return **400** with an **empty body**
- If i send you a **png file** with the jpg extension, you need to return a **png file too**

I give lot of freedoms on this project but don’t forget that a misconfigured docker image will lead you to 0/100 quickly.

You have until **next Wednesday 23:59** to finish your project.