NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

High Availability

Cloud Infrastructure Engineering

Nanyang Technological University & Skills Union - 2022/2023

Course Content

- Quick Check-In
- Dive into the basics of High Availability
- Explore the components needed for High Availability setup
- Calculate High Availability (3 Nines)
- Choose the right steps for High Availability

Time	What	How or Why
7:15pm - 7:25pm	Part 1 - Presentation	High Availability
7:25pm - 7:35pm	Part 2 - Presentation	Components for High Availability
7:35pm - 7:50pm	Part 3 - Presentation	Measures of High Availability
7:50pm - 8:10pm	Break + Activity	Measuring Availability
8:15pm - 8:25pm	Part 4 - Presentation	How to achieve High Availability
8:25pm - 8:35pm	Part 5 - Presentation	Best practices for High Availability
8:35pm - 9:00pm	Activity	Use case on achieving High Availability
9:00pm - 10:00pm	Assignment Briefing & Wrap Up	

Recap

- Server Clustering
  - Load Balancer
    - Application Load Balancer
    - Network Load Balancer
    - Gateway Load Balancer
- Whitelist
- Blacklist

High Availability

Let’s Cover Some Core Concepts

Resiliency - Ability of a storage system to **self-heal, recover, and continue operating** after **encountering failure, outage**, security incidents, etc

Reliability - Probability that the **storage system (hardware) will work as expected**

Durability - Continued **persistence** of data

What is Availability?

The term "availability" in computing refers to both the **amount of time that a service is accessible** and the amount of time it takes for a system to **react to a user's request.**

High availability refers to a property of a system or component that **guarantees a high level of operational performance for a specific amount of time**.

What is Availability?

An availability value of **100%** would mean that the system **NEVER fails.**

Availability is frequently expressed as a **percentage** indicating how much **uptime** is expected from a specific system or component in a given period of time.

For instance, a system that promises **99% uptime** over the course of a year may have up to **3.65 days of outage (1%)**.

Why is High Availability Important?

Whatever the cause, **downtimes can have negative impact** on the health of your company, including missed productivity, lost business opportunities, lost data, and impaired brand reputation.

As a result, IT staff continuously work to implement appropriate methods to **reduce downtime and guarantee continual system availability.**

Why is High Availability Important?

Maintain Service Level Agreements

For MSPs (Managed Service Providers) to guarantee the delivery of high-quality services to their clients, **maintaining uptime is a crucial requirement**.

High-availability systems enable MSPs to consistently meet their SLAs and **guarantee that their clients' networks don't go down**.

Why is High Availability Important?

Improve Customer Relations

Customers that experience **frequent business interruptions** due to outages may get **dissatisfied**.

High-availability environments **minimize the possibility of future downtime** and can assist MSPs in establishing enduring connections with clients by ensuring their satisfaction.

Why is High Availability Important?

Establish Company Reputation

System availability is a crucial sign of how well services are delivered.

Because of this, MSPs can use high-availability environments to preserve system uptime and establish a solid brand identity in the marketplace.

Think of companies like Shopee, Grab, Facebook vs Citi Mobile App, DBS App

Why is High Availability Important?

Improve Data Security

Having high availability can considerably **lower the likelihood** that crucial business data will be improperly accessed or stolen by decreasing the occurrence of system downtimes.

What Are The Components Needed for HA?

Factors of HA

For the actual implementation of high availability, a number of factors need to be properly taken into account.

High availability depends on elements other than just software implementation, including:

Environment/ Location

If all the servers are located in the same geographical area, an event such as an earthquake or flooding could **take the whole system down**.

Having **redundant servers in different datacenters and geographical areas** will increase reliability.

Hardware

Highly available servers should be **resilient to power outages and hardware failures**, including hard disks and network interfaces.

Software

The whole software stack, including the operating system and the application itself, must be **prepared for handling unexpected failure** that could potentially require a **system restart**.

Data

Data loss and inconsistency can be caused by several factors, and it’s not restricted to hard disk failures.

Highly available systems must **account for data safety** in the event of a failure.

Network

Unplanned network outages represent a possible point of failure for highly available systems.

It is then important that a **redundant network strategy** is in place for possible failures.

Summary

- Environment/ Location
- Hardware
- Software
- Data
- Network

Break Time

Availability Measures

Availability Measures

Even while three nines, or 99.9%, is typically thought to be a respectable uptime, it still equates to **8 hours and 45 minutes of downtime annually**.

Let's look at the table showing how the different availability levels translate into downtime hours.

Availability Measures

Activity

Let's calculate the **downtime per month in HOURS/MINUTES** and **downtime per week in HOURS/MINUTES**:

Availability %	Class of Nines	Downtime Per Year
99%	Two Nines	3.65 days
99.9%	Three Nines	8.77 hours
99.99%	Four Nines	52.60 minutes
99.999%	Five Nines	5.26 minutes

Availability Nines

bmc

The Nines of Availability

Availability percentages vs service downtime

Availability %	Downtime per year	Downtime per month	Downtime per week
90% (one nine)	36.5 days	72 hours	16.8 hours
99% (two nines)	3.65 days	7.20 hours	1.68 hours
99.5%	1.83 days	3.60 hours	50.4 minutes
99.9% (three nines)	8.76 hours	43.8 minutes	10.1 minutes
99.95%	4.38 hours	21.56 minutes	5.04 minutes
99.99% (four nines)	52.56 minutes	4.32 minutes	1.01 minutes
99.999% (five nines)	5.26 minutes	25.9 seconds	6.05 seconds
99.9999% (six nines)	31.5 seconds	2.59 seconds	0.605 seconds
99.99999% (seven nines)	3.15 seconds	0.259 seconds	0.0605 seconds

Break Time

Achieving HA

How Can We Achieve HA?

A system cannot become more stable or attain high availability by adding additional components.

In fact, it might have the reverse effect because **adding more components can make failures more likely.**

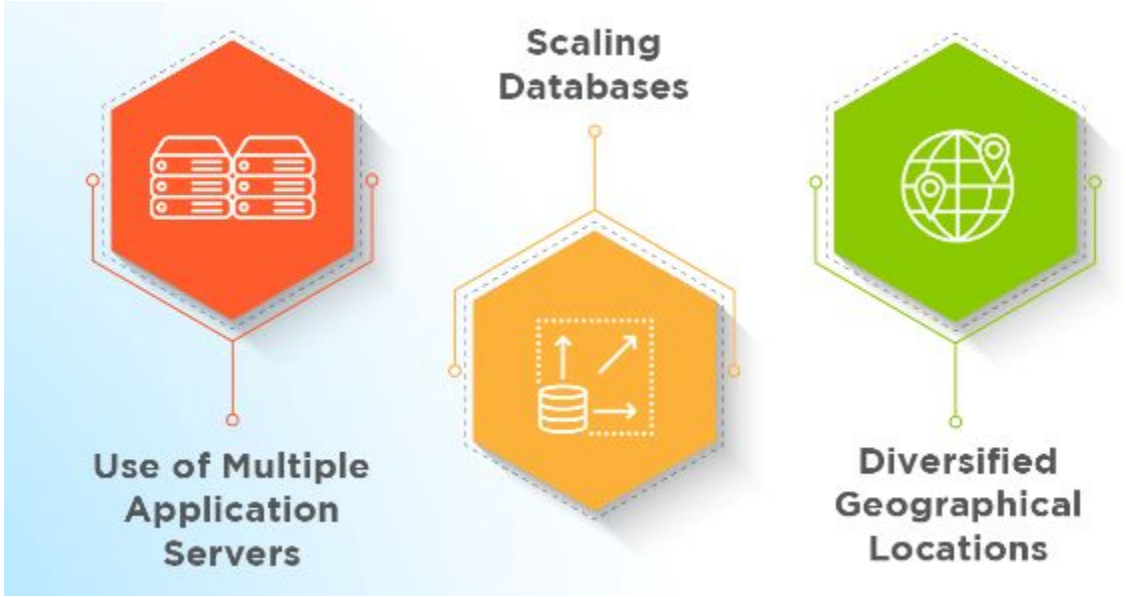
How Can We Achieve HA?

Modern designs enable the **distribution of workloads across numerous instances**, such as a network or a cluster, which aids in load balancing.

Load balancing is an approach to resource management that maximizes output, reduces reaction times, and avoids overloading any system.

Failover systems, which include switching to a backup server, component, or network when a current one fails, is also another aspect of this process.

How Can We Achieve HA?



Use Of Multiple Application Servers

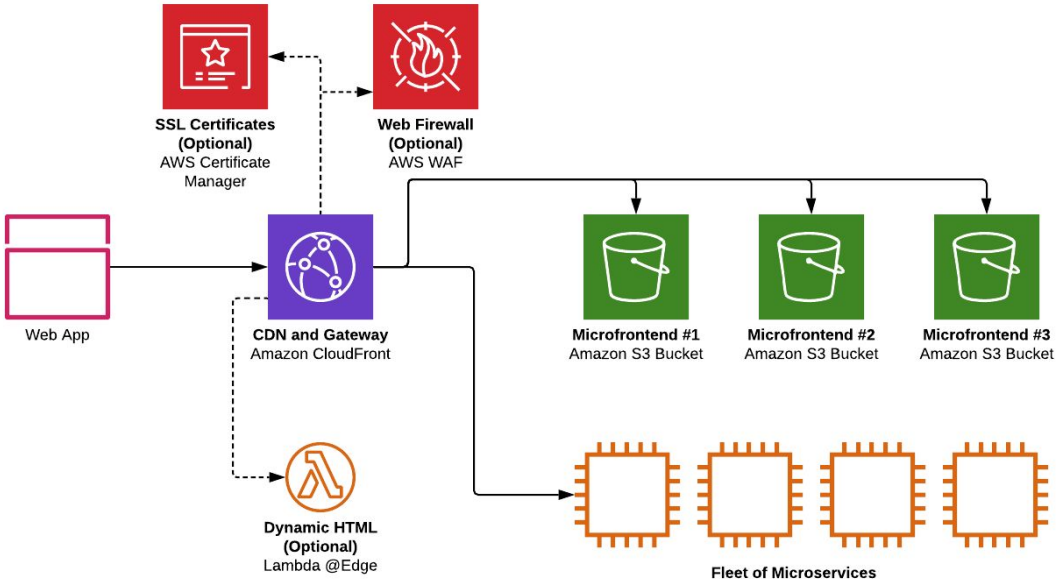
Deploying the application across numerous servers is a logical approach.

To ensure that **no single server is overloaded** and the **production is at its optimum**, the load must be balanced among all the servers.

Parts of the application can also be **deployed on different servers**.

For instance, there could be a separate server for processing static assets like photos or one for managing email (like a Content Delivery Network).

Use Of Multiple Application Servers



Scaling Database

Databases are just as crucial to services as application servers, it is necessary to keep in mind.

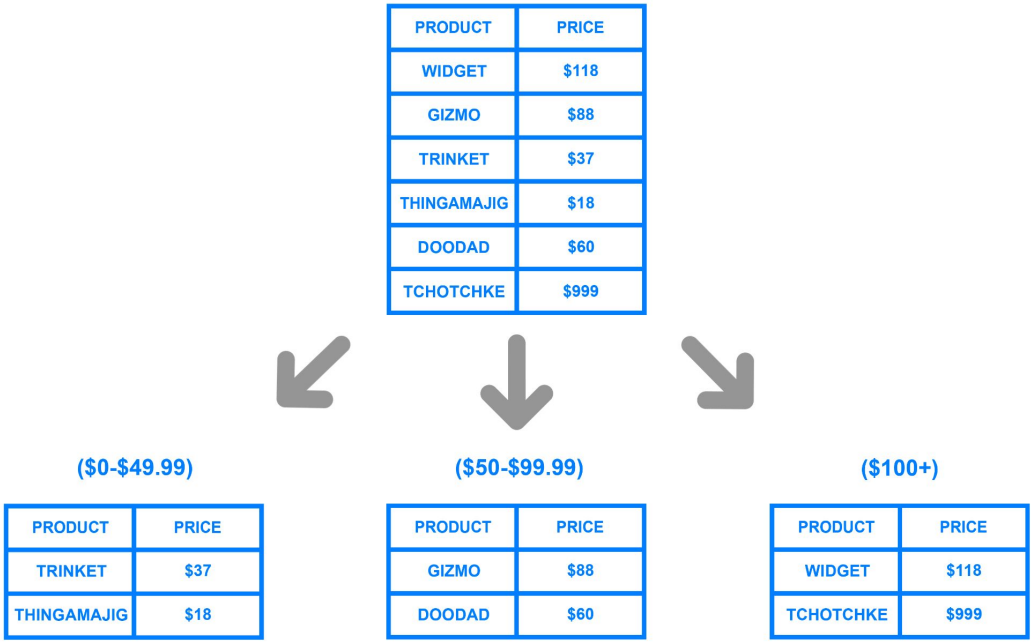
Databases operate on different servers and are also **prone to failure**. What's worse is that database crashes may result in **user data loss or client data**, which may be expensive and impact company's reputation.

Scaling Database

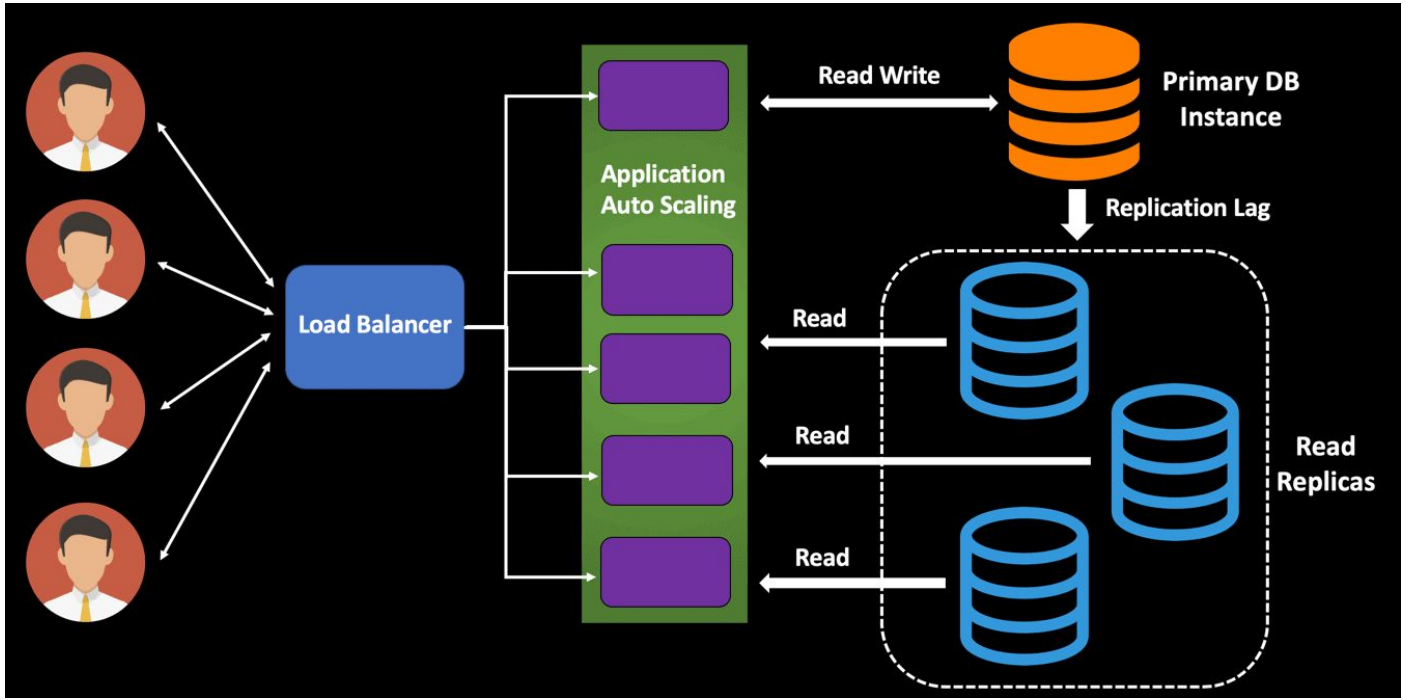
Redundancy is a process that **achieves failure detectability and prevents common cause failures** to provide systems with high levels of availability. Maintaining slave servers that can take over in the event that the primary server fails can help with this.

Sharding is a fascinating idea for scaling databases. A shard is a **horizontal database division** where rows from the same table are run on different servers.

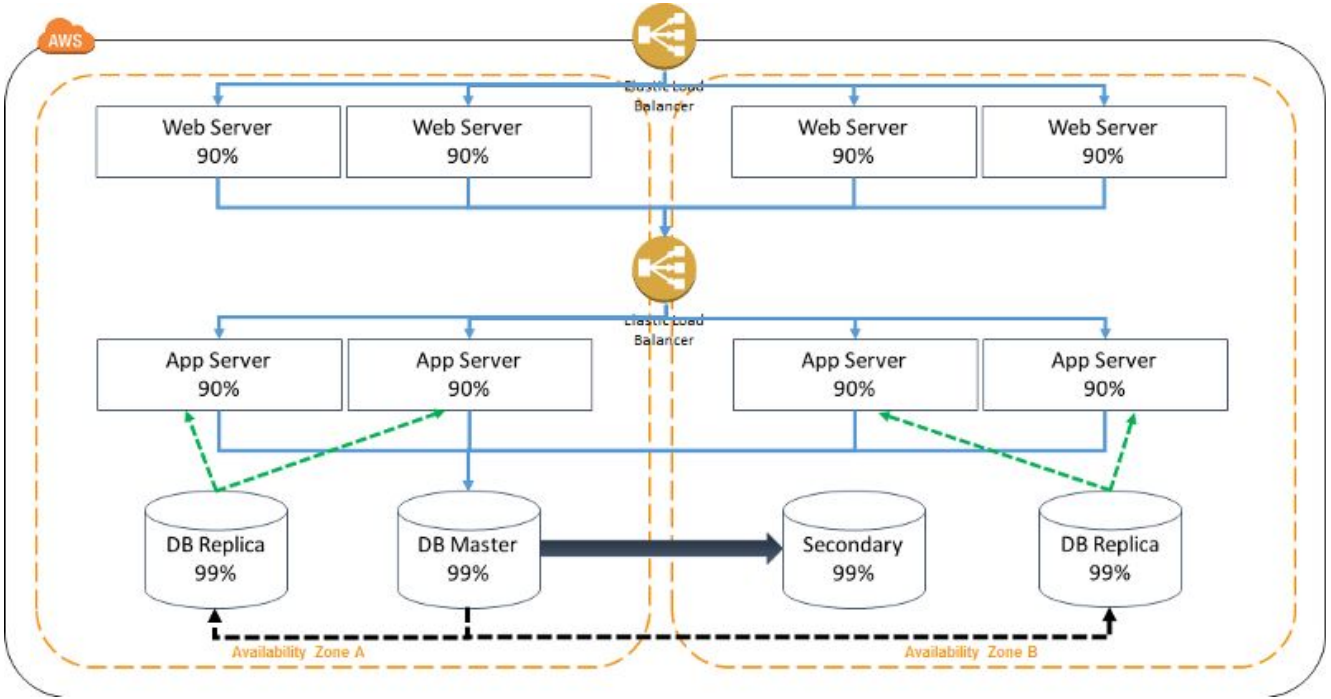
Scaling Database



Scaling Database



Scaling Database

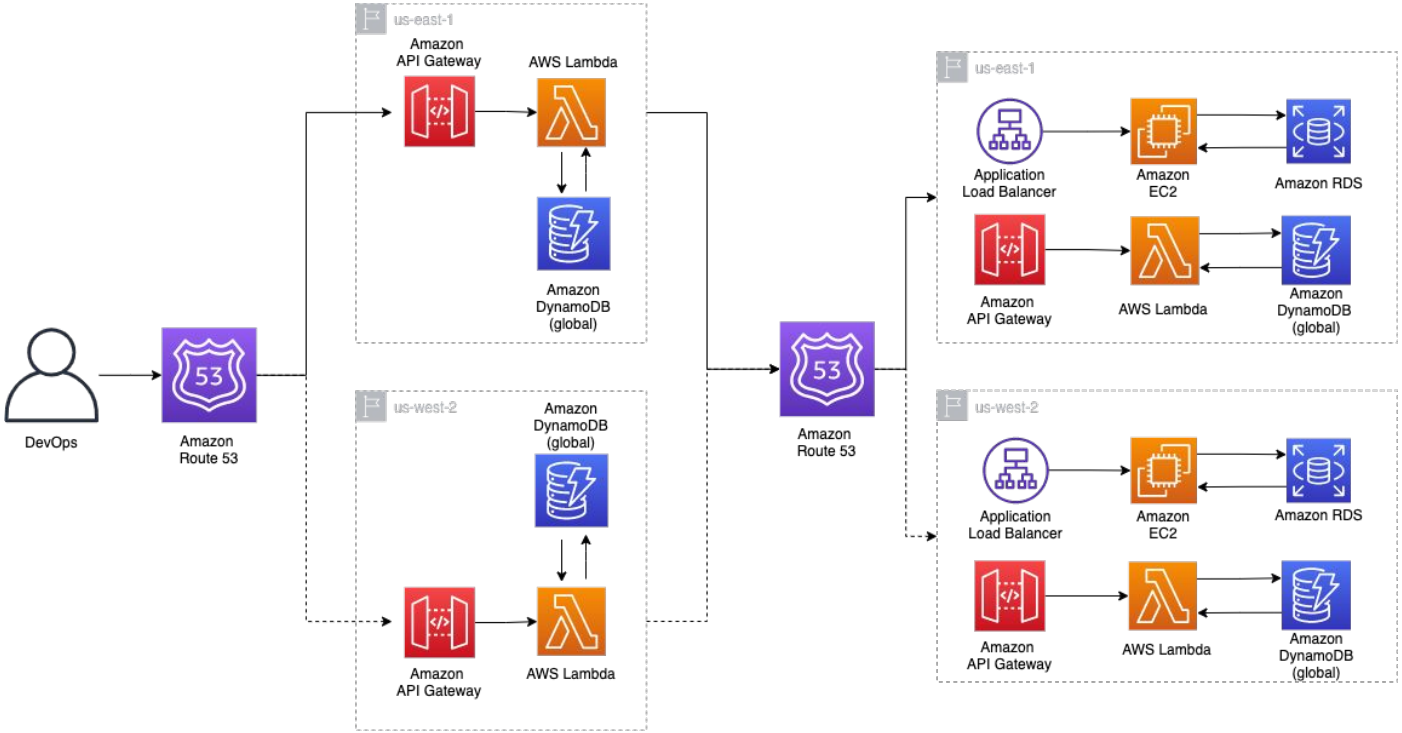


Diversified Geographic Locations

What happens if all of the servers are housed in the same data center and an event like a *natural disaster damages the data center* where the servers are kept? This may result in **lengthy downtimes**.

Servers must be **spread out in different places**. Choose the **appropriate physical locations of the servers** with the majority of contemporary web services by making intelligent decisions to ensure that the servers are spread out globally rather than localized in one place.

Diversified Geographic Locations



Best Practices For HA

Best Practices For HA

It is strongly advised, especially for **mission-critical applications**, to employ a High Availability (HA) design to r**educe system failures and prevent both planned and unplanned downtimes**.

Experts in availability maintain that **well-designed and thoroughly tested** components are essential for any system to be **highly available.**

Best Practices For HA

Data Backups, Recovery, and Replication

Never save valuable data without adequate backups, replication, or the capacity to recreate the data.

Every data center should make **early preparations** for data loss or corruption.

Making a **complete backup of the primary database and then incrementally checking the source server for data corruptions** is the suggested method for ensuring data integrity.

The first step in recovering from a catastrophic system loss is to create **comprehensive backups**.

Data Backups, Recovery, and Replication

In AWS, your options are:

1. Automated Backups
2. Point-in-time Restores
3. Database Snapshots
4. Database Snapshot Copies
5. Database Snapshot Sharing

https://aws.amazon.com/rds/features/backup/

Clustering

The goal of high availability is to **deliver application services** notwithstanding errors. In the event of a problem, clustering can offer immediate failover application services.

When a server goes down, a "cluster aware" application service can **fall back to a backup server** and still call resources from several servers.

Load Balancing

When instances of a server fail, the traffic is immediately **transferred to servers that are still up and running**, replacing the failed instances seamlessly.

Load balancing not only promotes high availability but also progressive **scalability**.

Higher levels of **fault tolerance** are made possible inside service applications.

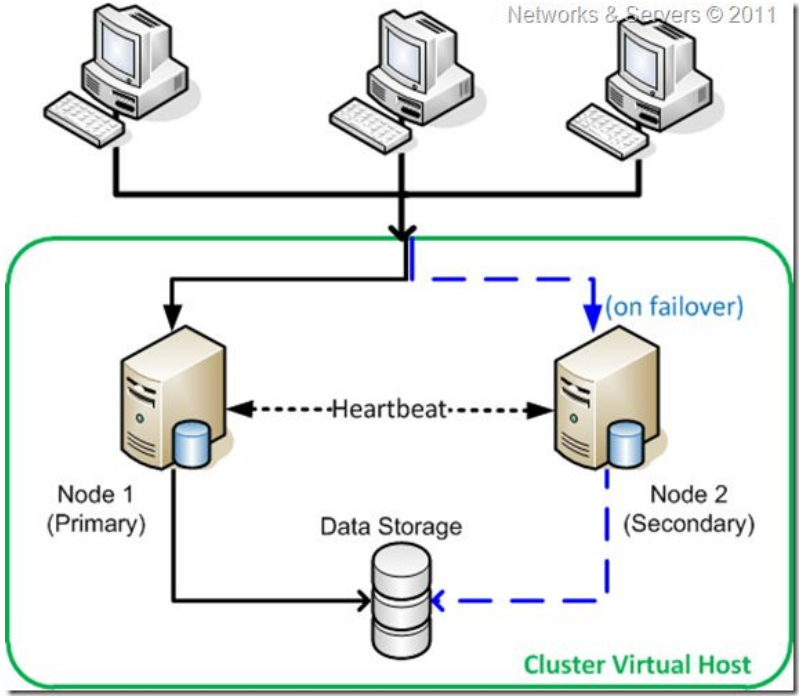
Failover Solutions

In a nutshell, failover is a backup operational mode where a **secondary system takes over a primary system's functions in the event that the original one goes down due to failure or planned downtime**.

When the backup server is only launched after the first one has been entirely shut down, this is known as a "cold failover."

When all the servers are active at the same time and just one server is handling the whole load, this is known as a "hot failover." Tasks are **automatically transferred from one scenario to the other** to keep the end user's experience as fluid as possible.

Failover Solutions



Geographic Redundancy

Multiple servers are installed at **geographically distant places**, just like in the case of geo-replication.

The places ought to be **dispersed internationally** rather of being localized in one place.

Running distinct application stacks in each site is essential so that if one location fails, the other can still function. These places should have **complete independence** from one another.

Plan For Failure

Data on resource usage or failure can be utilized in order to **identify issues and spot trends should be retained** by organizations.

Only by **continuously monitoring operational workload** can this data be gathered.

A **recovery plan** should be thoroughly written and periodically evaluated and tested.

Staff training on availability engineering can improve their skills in designing, deploying, and maintaining high availability architectures.

Security policies should also be put in place to **curb incidences of system outages** due to security breaches.

What’s Next?