**What is High Availability?**

In this lesson, you will learn about high availability and its importance in online services.

**We'll cover the following**

* + [What is high availability?](https://www.educative.io/module/lesson/web-application-architecture-101/gxprGvGwKY3#What-is-high-availability?)
  + [How important is high availability to online services?](https://www.educative.io/module/lesson/web-application-architecture-101/gxprGvGwKY3#How-important-is-high-availability-to-online-services?)

*Highly available* computing infrastructure is the norm in the computing industry today. When it comes to cloud platforms, it is their key feature that enables the workloads running on them to be *highly available*.

This lesson is an insight into high availability. It covers all the frequently asked questions about it, including:

* What is it?
* Why is it so important to businesses?
* What is a highly available cluster?
* How do cloud platforms ensure the high availability of the services running on them?
* What are fault tolerance and redundancy? How are they related to high availability?

So, without any further ado. Let’s get on with it.

**What is high availability?**[#](https://www.educative.io/module/lesson/web-application-architecture-101/gxprGvGwKY3#What-is-high-availability?)

*High availability*, also known as *HA*, is the ability of the system to stay online despite having failures at the infrastructural level in real-time.

High availability ensures the service’s uptime is much more than usual. It improves the reliability of the system and ensures minimum downtime.

The sole mission of highly available systems is to stay online and stay connected. A rudimentary real-world example of staying highly available is having backup generators to ensure continuous power supply in case of any power outages in hospitals, air traffic control towers and so on.

In the industry, *HA* is often expressed as a percentage. For instance, when the system is *99.99999%* highly available, it simply means *99.99999%* of the total hosting time the service will be up. You might often see this in the *Service Level Agreements* (SLA) of cloud platforms.

**How important is high availability to online services?**[#](https://www.educative.io/module/lesson/web-application-architecture-101/gxprGvGwKY3#How-important-is-high-availability-to-online-services?)

Social applications going down for a bit and then bouncing back might not impact businesses that much. However, there are mission-critical systems like aircraft systems, spacecrafts, mining machines, hospital servers, and finance stock-market systems that cannot afford to go down at any time. After all, lives depend on it.

The smooth functioning of the mission-critical systems relies on continual connectivity with their network/servers. These are the instances that do not work without super highly available infrastructures.

Besides, no service likes to go down, critical or not. To meet the high availability requirements, systems are designed to be *fault-tolerant* and their components are made *redundant*.

What are *fault-tolerant* and *redundancy* in distributed systems? I will discuss this up next.

**Reasons For System Failures**

In this lesson, we will understand the common reasons for system failure.

**We'll cover the following**

* + [Software crashes](https://www.educative.io/module/lesson/web-application-architecture-101/N8KVKRvGymK#Software-crashes)
  + [Hardware failures](https://www.educative.io/module/lesson/web-application-architecture-101/N8KVKRvGymK#Hardware-failures)
  + [Human errors](https://www.educative.io/module/lesson/web-application-architecture-101/N8KVKRvGymK#Human-errors)
  + [Planned downtime](https://www.educative.io/module/lesson/web-application-architecture-101/N8KVKRvGymK#Planned-downtime)

Before delving into the *HA* system design, *fault-tolerance*, and *redundancy*, we will first discuss the common reasons why systems fail.

**Software crashes**[#](https://www.educative.io/module/lesson/web-application-architecture-101/N8KVKRvGymK#Software-crashes)

I am sure you are pretty familiar with software crashes. Applications crash all the time, be it on a mobile phone or a desktop.

We also have to deal with corrupt software files. Remember the *BSOD* blue screen of death in Windows? *OS* crashing, memory-hogging unresponsive processes. Likewise, software running on cloud nodes crashes unpredictably and takes down the entire node.

**Hardware failures**[#](https://www.educative.io/module/lesson/web-application-architecture-101/N8KVKRvGymK#Hardware-failures)

Another reason for system failure is hardware crashes, including overloaded *CPU* and *RAM*, hard disk failures, nodes going down, and network outages.

**Human errors**[#](https://www.educative.io/module/lesson/web-application-architecture-101/N8KVKRvGymK#Human-errors)

This is the biggest reason for system failures. It includes flawed configurations and whatnot. Google made a tiny network configuration error, and it took down almost half of the internet in Japan. [An interesting read](https://thenextweb.com/google/2017/08/28/google-japan-internet-blackout/). <https://thenextweb.com/news/google-japan-internet-blackout>

**Planned downtime**[#](https://www.educative.io/module/lesson/web-application-architecture-101/N8KVKRvGymK#Planned-downtime)

Besides the unplanned crashes, there are planned downtimes that involve routine maintenance operations, patching software, hardware upgrades, etc.

These are the primary reasons for system failures. Now, let’s talk about how HA systems are designed to overcome these system downtime scenarios.

**Achieving High Availability - Fault Tolerance**

In this lesson, you will learn about fault tolerance and designing a HA fault-tolerant service.

**We'll cover the following**

* + [What is fault tolerance?](https://www.educative.io/module/lesson/web-application-architecture-101/RMlA1MPA7OY#What-is-fault-tolerance?)
  + [A highly available fault-tolerant service – Architecture](https://www.educative.io/module/lesson/web-application-architecture-101/RMlA1MPA7OY#A-highly-available-fault-tolerant-service-%E2%80%93-Architecture)

There are several approaches to achieve *HA*. The most important of them is to make the system fault-tolerant.

**What is fault tolerance?**[#](https://www.educative.io/module/lesson/web-application-architecture-101/RMlA1MPA7OY#What-is-fault-tolerance?)

*Fault tolerance* is a system’s ability to stay up despite taking hits.

A *fault-tolerant* system is equipped to handle faults. Being *fault-tolerant* is an essential element in designing life-critical systems.

In a fault-tolerant system, out of several *instances/nodes* running a service, a few go offline and bounce back all the time. In case of these internal failures, the system can work at a reduced level without going down entirely.

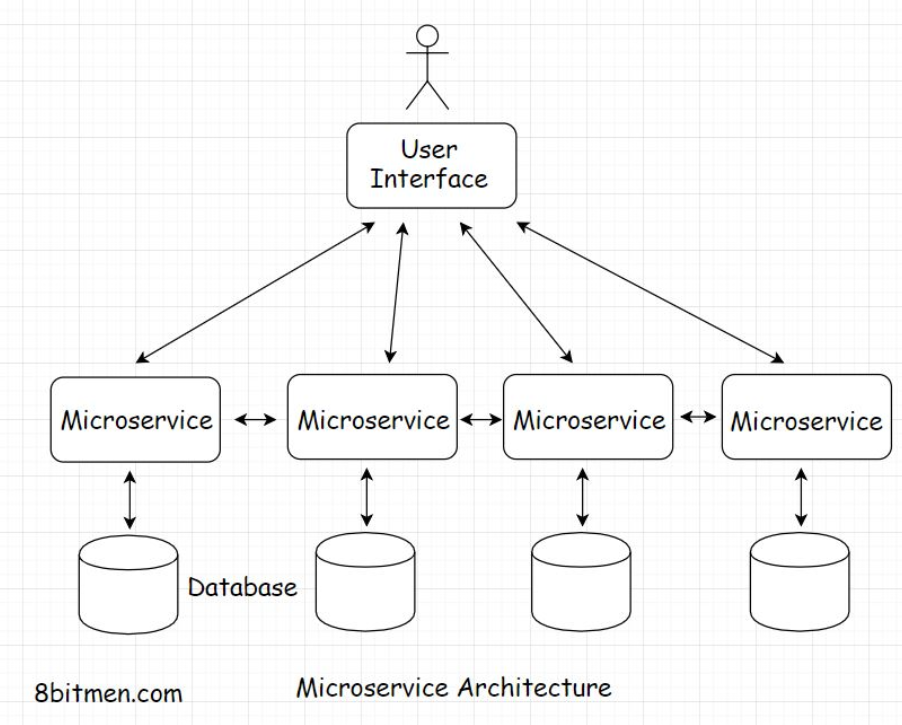
An elementary example of a system like this is social networking platforms. In the case of backend node failures, a few services of the app, such as image upload, post likes, etc., may stop working. However, the application as a whole will still be up. This approach technically is also known as *fail soft*.

**A highly available fault-tolerant service – Architecture**[#](https://www.educative.io/module/lesson/web-application-architecture-101/RMlA1MPA7OY#A-highly-available-fault-tolerant-service-%E2%80%93-Architecture)

A *fault-tolerant* system can be designed at the *application* and the *deployment level*. Since apps are typically deployed on the cloud in the modern web landscape, I discuss the deployment infrastructure in detail in my *cloud computing* course. It would be overkill to state everything in this one.

Also, this course, *Web Application and Software Architecture 101*, is part one of my *Zero to Application Architect* learning track, which enables you to be a pro at designing large-scale distributed services from knowing nothing. The *Cloud computing 101* course follows this course in the learning track. Future courses will be released soon. I’ll talk more about this track at the end of this course, though.

So, to achieve high availability at the application level, the entire massive service is architecturally broken down into more granular loosely coupled services called *microservices*.



There are many upsides of splitting a big monolith into several microservices:

* Easy management and maintenance
* Ease of development
* Ease of adding new features to a service without affecting other services
* Scalability and high availability of the system

Every *microservice* takes the onus of running different features of an application such as *photo upload, comment system, instant messaging, groups, marketplace*, etc. In this case, even if a few services go down, the other services of the application are still up.

# Redundancy

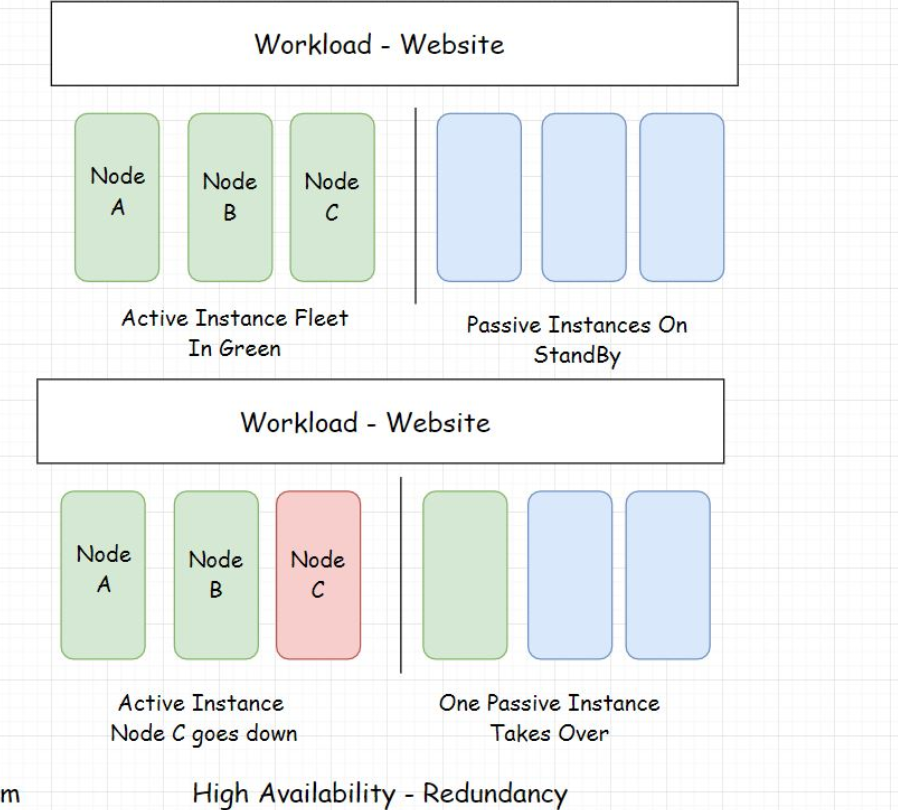
In this lesson, we will explore redundancy as a high availability mechanism.

**We'll cover the following**

* + [Redundancy – Active-passive HA mode](https://www.educative.io/module/lesson/web-application-architecture-101/R8XVYW3JoG0#Redundancy-%E2%80%93-Active-passive-HA-mode)
  + [Getting rid of single points of failure](https://www.educative.io/module/lesson/web-application-architecture-101/R8XVYW3JoG0#Getting-rid-of-single-points-of-failure)
  + [Monitoring and automation](https://www.educative.io/module/lesson/web-application-architecture-101/R8XVYW3JoG0#Monitoring-and-automation)

## Redundancy – Active-passive HA mode[#](https://www.educative.io/module/lesson/web-application-architecture-101/R8XVYW3JoG0#Redundancy-%E2%80%93-Active-passive-HA-mode)

Redundancy is duplicating the server instances and keeping them on standby to take over in case any of the active server instances go down. It is the fail-safe backup mechanism in the deployment infrastructure.



The illustration above shows the redundant instances. One of the redundant instances takes over when the active instance goes offline.

This instance setup approach is also known as the Active-passive HA mode. An initial set of nodes are active, and a set of redundant nodes are passive, on standby. Active nodes get replaced by passive nodes in case of failures.

There are systems like GPS, aircraft, communication satellites, etc., that have zero downtime. The availability of these systems is ensured by making the components redundant.

## Getting rid of single points of failure[#](https://www.educative.io/module/lesson/web-application-architecture-101/R8XVYW3JoG0#Getting-rid-of-single-points-of-failure)

Distributed systems became mainstream with large-scale applications solely because, in them, we can eliminate the single points of failure that were a big downside of a monolithic architecture. I discussed this in the previous lesson how microservices facilitate a fault-tolerant architecture.

In a highly available system, a large number of distributed server nodes work in conjunction with each other to achieve a single synchronous application state.

When so many redundant nodes are deployed, there are no single points of failure in the system. In case a node goes down, redundant nodes take its place. The system as a whole remains unimpacted.

Single points of failure at the application component level mean bottlenecks. I discussed this earlier, having just one database instance handling requests from a number of application nodes. We should detect bottlenecks in performance testing and get rid of them as soon as possible.

## Monitoring and automation[#](https://www.educative.io/module/lesson/web-application-architecture-101/R8XVYW3JoG0#Monitoring-and-automation)

Systems should be well monitored in real-time to detect any bottlenecks or single point of failures. Automation enables the instances to self-recover without any human intervention. It gives the instances the power of self-healing.

Also, the systems become intelligent enough to add or remove instances on the fly as per the requirements. Kubernetes is one good example of this.

Since the most common cause of failures is human error, automation helps cut down failures considerably.

# Replication

In this lesson, you will learn about replication as a high availability mechanism.

**We'll cover the following**

* + [Replication – Active-active HA mode](https://www.educative.io/module/lesson/web-application-architecture-101/JQBDy2561yD#Replication-%E2%80%93-Active-active-HA-mode)
  + [Geographical distribution of workload](https://www.educative.io/module/lesson/web-application-architecture-101/JQBDy2561yD#Geographical-distribution-of-workload)

## Replication – Active-active HA mode[#](https://www.educative.io/module/lesson/web-application-architecture-101/JQBDy2561yD#Replication-%E2%80%93-Active-active-HA-mode)

Replication means having a number of similar nodes running the workload together. There are no standby or passive instances. When a single or a few nodes go down, the remaining nodes bear the load of the service.



This approach is also known as the Active-active high availability mode. In this approach, all the server instances are active at any point in time.

## Geographical distribution of workload[#](https://www.educative.io/module/lesson/web-application-architecture-101/JQBDy2561yD#Geographical-distribution-of-workload)

As a contingency for natural disasters, regional power outages, and other big-scale failures, data center workloads are spread across different data centers across the world in different geographical locations.

This eliminates a single point of failure in the context of data center zones. If a natural disaster wipes out a few data centers in a certain zone, other data centers in different geographical zones are still powering the application. Also, the latency is reduced significantly due to the proximity of data to the user due to the global replication of data centers.

All highly-available fault-tolerant design decisions are subjective to how critical the system is? The odds of components failing and so on.

Businesses often use multi-cloud platforms to deploy their workloads which ensures further availability. If things go south with one cloud provider, they have another to fail back over.

# High Availability Clustering

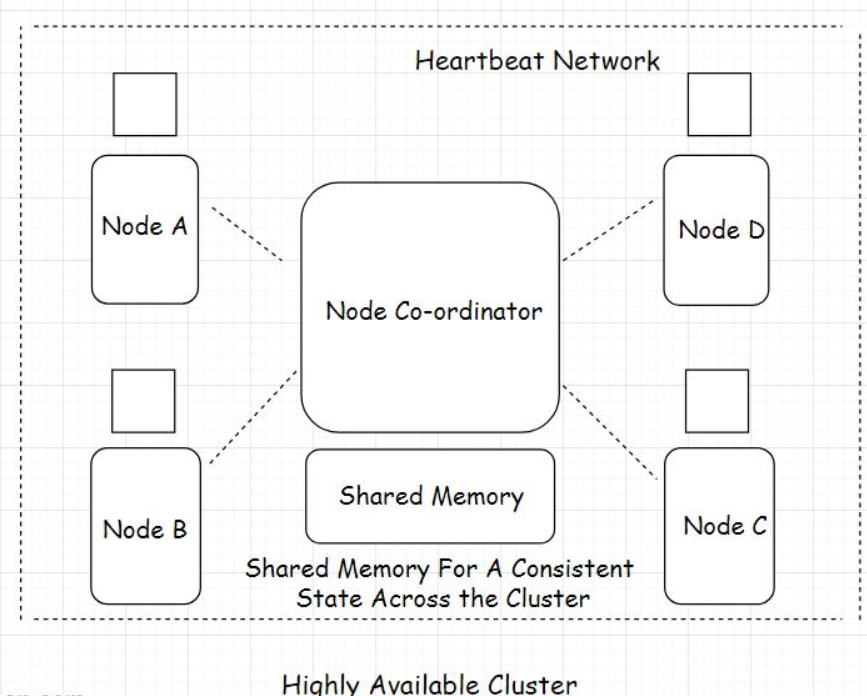
In this lesson, you will cover high availability clustering.

Now that we have a clear understanding of high availability, let’s talk a bit about the high-availability cluster.

A high availability cluster, also known as the fail-over cluster, contains a set of nodes running in conjunction with each other that ensures the high availability of the service.

The nodes in the cluster are connected by a private network called the heartbeat network that continuously monitors the health and the status of each node in the cluster.

A single state across all the nodes in a cluster is achieved with the help of a shared distributed memory and a distributed coordination service like the Zookeeper.



To ensure availability, HA clusters use several techniques such as disk mirroring/Redundant Array of Independent Disks (RAID), redundant network connections, redundant electrical power, etc. All the possible components having a single point of failure are made redundant to ensure the availability of the service.

Multiple HA clusters run together in one geographical zone ensuring minimum downtime and continual service. If you wish to delve more into clustering please go through my cloud course.

Alright, Folks! So now we have a pretty good understanding of scalability and high availability. These two concepts are crucial to software system design.

Let’s move on to the next chapter, where we discuss load balancing in web applications.