Redundancy

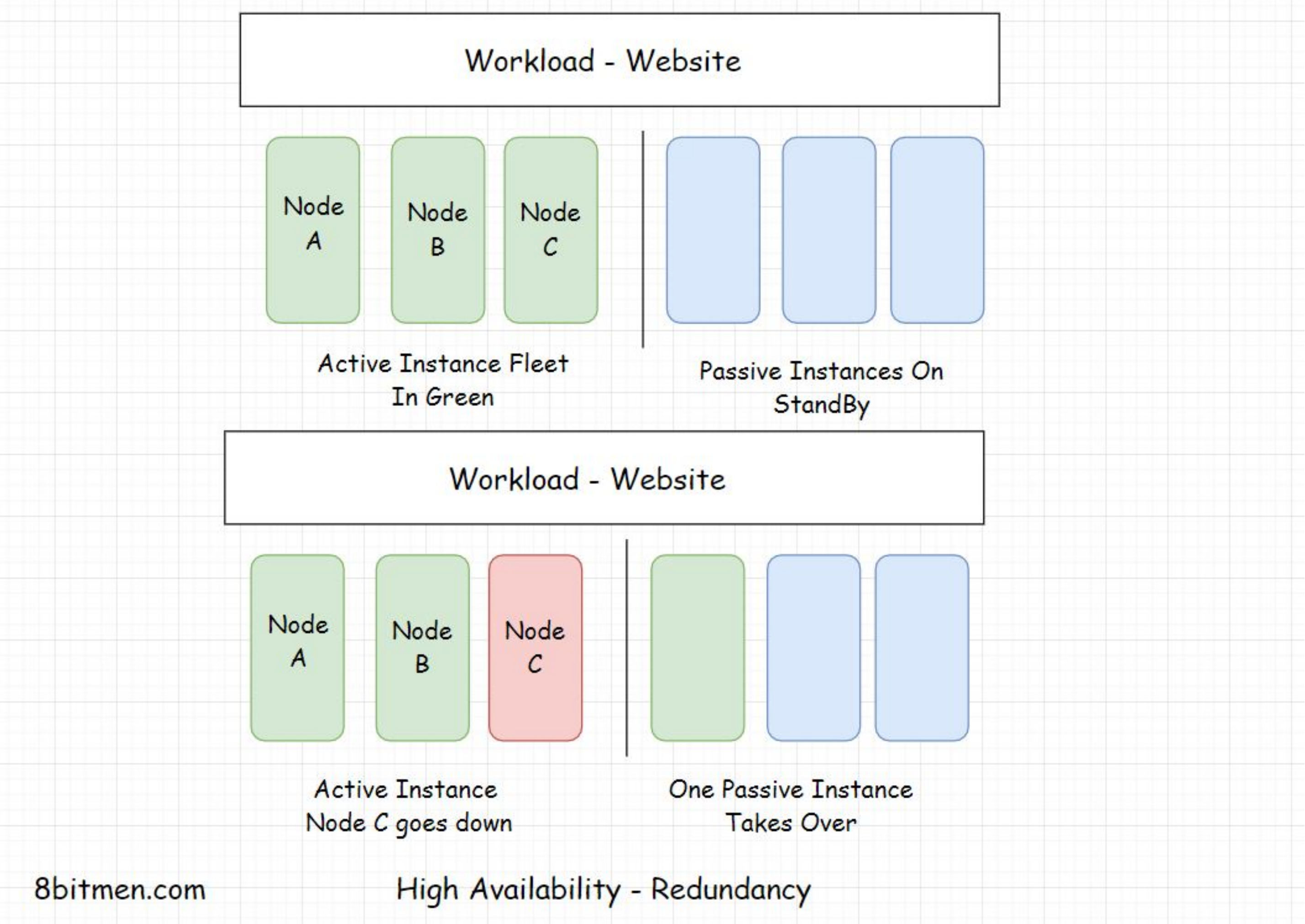
In this lesson, we discuss redundancy as a high availability mechanism.

We'll cover the following

- Redundancy – Active-passive HA mode
- Getting rid of single points of failure
- Monitoring and automation

Redundancy – Active-passive HA mode

Redundancy is duplicating the components or instances and keeping them on standby to take over in case the active instances go down. It's the fail-safe, backup mechanism.


Workload - Website
Node A
Node B
Node C
Active Instance Fleet In Green
Passive Instances On StandBy
Workload - Website
Node A
Node B
Node C
Active Instance Node C goes down
One Passive Instance Takes Over
8bitmen.com
High Availability - Redundancy

In the above diagram, you can see the instances active and on standby. The standby instances take over in case any of the active instances goes down.

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

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

Getting rid of single points of failure

Distributed systems became so popular because we can use them to get rid of only the points of failure present in a monolithic architecture.

A large number of distributed 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, and the system as a whole remains unaffected.

Single points of failure at the application level mean bottlenecks. We should detect bottlenecks in performance testing and get rid of them as soon as we can.

Monitoring and automation

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

Furthermore, the systems become intelligent enough to add or remove instances on the fly as per the requirements. Since the most common cause of failures is human error, automation cuts down failures significantly.

Let's cover the *replication* mechanism in the next lesson.