How to Build and Test High Availability IT Platforms

The terms scalability and high availability (HA) have never been more popular than now when there is a rising demand for stable and performant infrastructures designed to serve critical systems. While rising system load is a common concern, reducing downtime and eliminating single points of failure are equally critical. High availability is a characteristic of large-scale infrastructure design that takes these factors into account.

What Is High Availability?

High availability is often synonymous with high-availability systems, high-availability environments, or high-availability servers. In simple terms, when some of your IT infrastructure's components fail, high availability allows the system to keep it functioning.

High availability is vital for mission-critical systems since a service disruption might harm the business, leading to increased expenses or financial losses. Although high availability may not completely remove the risk of system downtime, it does ensure that the IT team has taken all required precautions to assure continuity of business operations.

Why Is High Availability Important?

Today, downtime and disruptions equate to income loss. From a business standpoint, high availability has become too critical. Customers get frustrated when services go down, which may lead loyal customers to seek out alternatives and use competitor services.

IT teams are always working to reduce downtime and assure system availability at all times. Downtime can cause a wide range of implications, including lost productivity, missed business opportunities, lost data, and a damaged brand image.

Businesses value high availability because it makes their services more reliable. Unexpected situations can cause even the most reliable systems and servers to fail. Therefore, it's critical to use high availability to reduce service interruptions, outages, and downtime. Highly available systems can recover from data loss and server failures automatically.

One of the many reasons for high availability is to avoid downtime. The following are some of the other reasons:

- **Ensuring data security** - By reducing system downtime through high availability, you can dramatically reduce the likelihood of your essential business data being accessed or acquired illegally.

- **Managing SLAs** – Maintaining uptime is a must for managed service providers who want to provide high-quality service to their customers. Managed service providers can use high-availability solutions to ensure that they meet their SLAs 100 percent of the time and that their clients' networks do not go down.

- **Maintaining brand reputation** - System availability is a key sign of your service quality. As a result, managed service providers can take advantage of high-availability environments to ensure system uptime and establish a strong brand in the market.

What Makes a System Highly Available?

The elimination of failure single points in your infrastructure is one of the high availability goals. A single point of failure is a component in your technology stack that, if unavailable, would cause a service outage. As a result, any component that is required for your application's correct functionality but lacks redundancy is deemed a single point of failure.

1.No Single Points of Failure

Each layer of your stack should be prepared for redundancy to eliminate single points of failure. Consider the following scenario: you have a load balancer and two identical, redundant web servers. Client traffic will be evenly spread among the web servers, but if one of them goes down, the load balancer will reroute all traffic to the remaining online server.

In this situation, the web server layer is not a single point of failure because redundant components for the same task are available. The load balancer, which sits on top of this layer, can detect component failures and change its behavior to ensure a quick recovery.

2.Hardware Redundancy

Building two or more computing systems or physical copies of a hardware component can ensure hardware redundancy. A system could have redundant servers, power supply, memory, and other components. You can lessen the risk of outage in the event of higher loads by using redundancy on high-level components.

3.Reliable Crossover

When a server breaks or stops responding, the reliable crossover must be put up to ensure that the failover systems take over. This is another component of implementing redundancy into HA systems. It allows a backup component to replace a failed component. Reliable crossover is the process of properly switching from one component to another without losing any data or affecting performance.

4.Software and Application Redundancy

Software and application redundancy functions similarly to hardware redundancy in that it fulfills the same function by running various pieces if another instance is affected. In high availability, fault tolerance, and reliability engineering, it is a critical concept. It also includes self-healing programs. Applying redundancy guarantees that you meet reliability goals while staying within technological restrictions.

5.Data Redundancy

Data redundancy is ensured via a high availability system, which means the same data is stored in multiple locations. This lowers the risk of data loss and assures that the data can be restored if one of the memory locations or servers fails. It also allows for the rectification of inaccuracies in data that has been transferred or stored.

6.Self-Monitoring for Failure

High availability systems contain self-healing and self-monitoring capabilities that can identify abnormal failure rates or afflicted instances. It ensures that the error is spotted and rectified promptly, with minimal impact on the system's functionality. Your uptime will increase as your self-monitoring feature becomes more efficient.

SCAND Best Practices

The analysis of needs in terms of processes is the first step to build a high availability system. This includes identifying the key processes, types of data they interact with, how data should be moved, stored, and its retention period.

The first step in system design is the definition of domains. A domain identifies a single service or a subset of a service. Then CPU and memory-intensive tasks are determined, necessary communications and integrations with third-party systems defined. It's also identified where synchronous and asynchronous communication will take place. The system architecture is being created with hosting provider capabilities in mind. It's being debated whether cloud-native services or a third-party vendor solution (for example, Kafka vs Google Pub/Sub) would be a superior choice.

The minimal architecture capable to deal with high availability should consist of a set of duplicated nodes/services (at least two copies, as required by PCI-DSS), a load balancer to route traffic to the nodes, a database with failover, and a monitoring facility capable of detecting problems with the environment.

Message queues and message queue brokers are used when direct synchronous HTTP communication between services is insufficient and data processing is distributed. By delegating messages per service replica, MQ makes it easier to deal correctly with competitive resource access. When event streaming is used, MQ brokers assist in the development of a robust distributed system. This allows operations such as logging, monitoring, and analytics collecting in settings to be clearly separated. It also enables all subscription services to maintain track of the most up-to-date sections of the data they require and to create scalable processing clusters.

Depending on project needs, fast memory storage can be introduced like Redis, Memcached to be able to keep fast access to key-value attributes like tokens or rendered UI portions and JSON data structures.

Kubernetes technology allows monitoring and orchestrating containers in zero-trust environments. It helps to detect broken containers and replace them in a matter of minutes, helps to horizontally scale with new replicas of the services helping to keep load spread between stateless clones.

The database is created with a failover being able to replace the primary instance promptly on the hardware level in case of failure. To be able to work in high load conditions, we create read-only replicas of a primary write-only database. Data processing is divided into write operations which are dealing with primary DB and query operations that are working with multiple database clones (CRQS). In case of failure of the primary database, one of the replicas becomes a primary one. In case of failure of a read replica, a new one is created and synced to be up-to-date using some time interval before going live.

To work with a higher load identified by PCI-DSS level 1 (>6 million transactions per year), processing clusters are created which work with their unique data partitions (their own services and databases). Data cluster is identified by the incoming request parameters (or by user token) on fronting the high-efficient API gateway layer. Consuming processes such as statistics collection and report generation are carried out in parallel in separate clusters.

Monitoring of the services' availability is done either by inbuilt cloud hosting means or standalone pro solutions like TICK, ELK stacks that allow to gather big portions of logs, metrics and use different notification channels to trigger incidents to the tech support team (email, SMS, slack, telegram, etc.).

Development teams work on services' repositories, branches of which are used to update the development/stage/production environment manually or automatically by commits (optionally with tags). The main development principle is to keep the CVS branch working and stable (CI). Before moving to the artifact building stage, unit/regression/automation/stress/security tests are run in a pipeline. If the system doesn't meet metrics identified with stress tests, delivery is canceled and returned to developers. Stress tests run in the CI/CD pipeline and consist of scenarios written using the following wide-spread tools: JMeter, artillery.io, k6, Gatling, and others.

We provide a seamless system update with no downtime using CD principles. We recommend the following deployment strategies: rolling, in which new versions of containers are steadily run and old versions are gradually replaced; canary, in which new version rollout is controlled for a subset of users and rolled out entirely once tests are completed (check out our software quality assurance services). Delivery can be done manually or automatically. Depending on the complexity of database migrations, it can be done immediately or gradually.

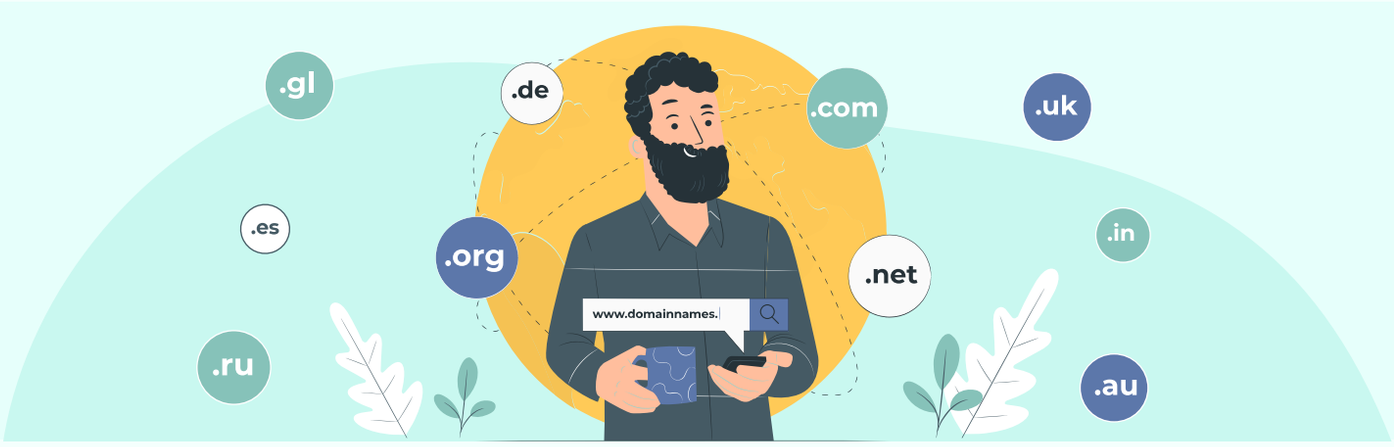
The most important thing about HA is to make sure all technical means are still working and protecting you from risks. Under regulations or not, our customers need to have policies for HA maintenance.

Under our provision the following approach is usually implemented:

- a policy on how the team should act within different infrastructure disaster scenarios;
- a policy that states how often, what team should hold HA tests; how the process should be documented, and what/how notifications should be maintained;
- disaster recovery tests;
- recovery training.

In short, we make sure that disasters regarding infrastructures are well-known, described and the operational team knows what to do to fall under SLA requirements.

Testing is done by simulating different outage scenarios: on service level, DB level, domain level, etc. Testing sessions are held for each component or the whole platform at once. A short-lived environment is used for this purpose.



Summary

High availability is no longer a luxury in today's competitive market. Failure of critical IT systems can result in significant costs for the company, ranging from decreased user productivity to a loss of income and trust from customers. As a result, IT and business leaders in small and medium companies must make high availability a priority for core applications.

A high-availability infrastructure is made up of hardware, software, and applications that are designed to recover fast in the event of a failure and sustain functionality with little downtime. A company with a well-articulated set of high availability best practices, including high availability analysis frameworks, business drivers, and system capabilities, will have better operational resilience and business agility.

Feel free to contact us and ask any questions: info@scand.com