# SRE GUIDE FOR QUICK RECOVERY

**Nazmuddin F Shaikh**

*Director of Engineering, Sam club Technologies, Walmart Inc, Bentonville, USA,*

***Email: s.nazmuddin@gmail.com***

**ABSTRACT**

SRE team or Site Resolution engineer is generally the team responsible for availability, latency, performance, efficiency, change management, monitoring, emergency response, and capacity planning. In today's cloud infrastructure and microservices architecture it has become more critical of a role with growing focus on the availability and performance. In most of the cases in today's world the SRE teams are the ones who even take up the responsibility for handling the recovery and RCA (Root cause analysis) for the issues and troubleshooting for the issues and handling the escalation matrix. Many operations teams today have a similar role, sometimes without some of the bits that I've identified. We are trying to have the focus on the quick steps which can be handy and can be used irrespective of the application and infrastructure.

**Keywords**: SRE, RCA, MTTR, MMTD, Availability, DR (Disaster Receovery)

## INTRODUCTION

The quick steps below are focused to talk about the recovery of any software application or site in general. It must be noted that the actual recovery time for any software application might be less, but the impact can be long lasting and never returning in terms of Users experience and trust. In worst case scenarios these can be during the High potential sales events and like Thanksgiving and Christmas which will even lead to loss of trust from other Business partners and leaders.

**Preconditions:**

When make sure we have the right documentation and clear understanding of the application which can be used in case of any failure or disaster. We talk about all the preparation and precautions, we can take to make sure such an event is not happening on any day, we should always be ready and have a clear plan of execution for such a disaster. In this short note we plan to discuss some basics steps which can be used to prepare our applications for recovery from such an unplanned disaster. The below are few of the key Basic resiliency needs for every application which we should have in place to

1) Should have a block diagram view of the services/applications involved in the overall workflow and black box each application/component.

Architecture review and services documentation should be mandated to be updated as part of each change for all critical applications.

2) Each component should define their Golden signals and should have a clear trend view of the Golden signals. Every new client onboarding and dependency injection should be validated to have proper golden signals built whenever a change is implemented.
3) Each application or component should have a DR playbook or well-defined failover strategy which should be executable in fraction of time.

   This playbook should be kept updated with each change that is implemented in the system at the design level. This should part of the change management and approval process for critical applications.
4) In general, critical application should have redundant routes for achieving a higher degree of availability. If one systems probability of failure is $1 - \alpha$, then the probability of the system being down which has 2 redundant subsystems either in different regions, clusters or data centers decreases by the below formula, $F = (1-\alpha 1) \times (1-\alpha 2)$.

   Availability = 1-F.

   So, a system having 2 regions with each 99% availability, the availability of the overall system theoretically goes to 99.99%. Each one can fail up to 1% and the probability that they both fail is $(1-99\%) \times (1-99\%) = .01\%$. This makes the availability of a. system using 2 regions or clusters as 99.99%.

**SRE metrics:**

Some of the most widely used terms and metrics determining the SRE functioning and metrics to measure the performance industry wide are as below.

**What Is MTTD?**

Mean Time to Detect (MTTD) is the average time it takes a team to discover a security threat or incident. This is general terms is the time the team took even to acknowledge there is an issue. This can vary from application to application and also it has its own calculation implications based on the deviation from normal trend etc.

**What Is MTTR?**

Mean Time to Respond (MTTR) measures the average time it takes to control and remediate a threat. This is the time which determines your responsiveness and alertness to handle any issue. This is where all your golden signals and alerting helps you.

**How Do You Measure MTTD and MTTR?**

MTTD and MTTR depend on a number of factors, including the size and complexity of your network, the size and expertise of your IT staff, your industry, and more. Another thing to keep in mind is that different companies measure things in different ways.

There are no industry-standard approaches to measuring between these two performance indicators, so granular comparisons between organizations can be problematic apples-vs-oranges affairs.

**Troubleshooting Steps:**

As we talk about the preconditions which we expect we should always have for an application, now let's try to deep dive into some specific course of action items which can be taken when we find ourselves in such a situation.

Initial reaction or Triage: Whenever any issue is reported, don't go with history and presumptions rather take a minute to look at the exact impact being noticed. Generally monitoring and alerting frameworks tend to go with trend charts and static monitors and alerts which can be faulty at times but nevertheless each instance should be looked at as a potential P0 to begin with.

**Top-Down approach:**

When looking at any issue, don't go with a bottom-up approach or start looking at each component one by one. That can be done by individuals once you have established the application being impacted Root cause or path to recovery.


Ideally you should start at this step, the other black box view and golden signals for each black box should be available with each team and should be always handy to work on.

Start looking at Top-Down approach and find which application in the funnel is not behaving as expected. Generally, if a core downstream is impacted, your metrics for all the funnel shows the impact.

**Isolate the Faulty:**

Try to isolate the faulty system or component as soon as possible and focus on recovery. While isolating the application or component don't get overridden by confidence or historical behavior of any application, rather take into consideration any and every component or application can fail. We have seen prejudice about a particular component or false confidence causes a lot of precious time to go waste. Don't try to find in detail why a particular System or component is failing and how we can fix the Root because of it. Try to find What is failing and what can be

easiest mitigation. Look at each Blocks Golden signals with an approach that anything can fail, never take any application or component assumed to be always working and to be resilient.

- Look at the Golden Signals in 3 different Perspective.
- The incoming requests or the Upstream applications are impacted.
- The downstream is impacted.
- System under observation is impacted.

For the Upstream being impacted, your incoming request trend and error trend in your application can serve the high-level view and help in deciding the Upstream status.

In most cases your request trend might also be impacted from the Retries by user, refresh, system retries etc. Also, just an increase in the Request should be handled by the current matured scalable systems with auto scale etc. The Request trend with a corresponding increase in Processing error in the system can mean it can be a data issue in the system or the request data also. Log analysis plays a critical role as well in this case as the upstream request flow might be impacted in a particular region and will show as an error in downstream processing in a particular region. Don't finalize the Root cause as upstream just based on Request trend.

For the downstream systems, always have your golden signals configured for latency, error trend and denials. In general, Downstream golden signals are clear and should help in deciding the impacted downstream. In some cases, if all downstream are showing impact and metrics for them are deviated. It can be your system calling all those dependencies or network in between.

**Black Box each component:**

Treat each application as a Black Box till you isolate the faulty system. Current System under examination is more complex. Treat your application as a black box to start with. Don't look at logs to start with and try to find what is happening in detail. Look at things from a high-level availability perspective.

For system we should have below params to look at.

1) System trends of CPU, Memory, processing time, threads, GC, etc.
2) Request processing trend of 2xx,5xx,4xx etc.

These two ways to look at each application will help in deciding that is it available or not.

Once you cover each block in a top-down approach and identify the system /application which is not working as expected, plan to mitigate the issue in parallel rather than deep diving into the RCA.

**Route to Mitigation.**

1) Mitigation becomes easier if you can have golden signals isolated by Region/Datacenters /Clusters etc.
2) Mitigation can be easily performed if region affinity, same DC calls, local cluster affinity is built in the system.
3) Take traffic routing, out of rotation decision based on the golden signals.
4) Disaster recovery, Failover strategy to passive application, etc. should be carried out if the Golden signals indicate problem is isolated.

**Root cause analysis:**

Any issue can happen because of two things.

A) A potential change in system or data.

B) A pile up effect or exhaustion of resources over a period, you just happen to cross the bottleneck at a moment.

C) A network change also influences the system and becomes difficult to identify.

So while trying to find the Root cause of any issue, go through the change list across the application. Change in system, change in data, change in infrastructure, network, monitoring, logging, etc. This will help to focus on the right area. The application configuration might have a retry, timeout setup, configuration etc. which might be kicking in when a certain data condition or system condition is reached, make sure you go through these flags, timeouts, retries etc. While doing RCA, try to compare trend charts and not only a snapshot of the impacted window. The difference can be used to nail down the Root cause. For RCA we should have the trend charts in detail, if possible, at a cluster or node level segregation, so that any issue at granular level can be looked at.

## CONCLUSION

While SRE teams can focus on different RCA and different matrix to identify the root cause and achieve best in class Availability and Performance for the applications. Troubleshooting is often a skill associated to the algorithm of thinking applied by a person or a team. By using simple steps listed above we can standardize and over a period refine those as per the need of the particular application to achieve best MTTD and MTTR. With cloud infrastructure at dispose scalability has become easy, but comes with overburden of cost and mismanaged resources, but it can be leveraged to attain better availability and Disaster recovery mode.

**References**:

1. N Kavyashree, MC Supriya, MR Lokesh, Site reliability engineering for IOS mobile application in small-medium scale industries

2. V Ukis , Establishing SRE Foundations: A Step-by-Step Guide to Introducing Site Reliability Engineering in Software Delivery Organizations

3. Betsy Bayers , Chris jones Site reliability engineering , oreilly books.

4. Laura Nolan,Seeing Like an SRE: Site Reliability Engineering as High Modernism

5. E Zio, Reliability engineering: Old problems and new challenges

6. E Zio ,Some Challenges and Opportunities in Reliability Engineering, IEEE, Volume: 65 Issue: 4

7. Nazmuddin Shaikh , Omni channel commerce and Generative AI.