Notes - Organizational Development - Technology - Part 2

- Dr Nick Hayward

A brief introduction to technology choices and structure with example usage for software projects.

Contents

Intro

Technology choices, both initial and ongoing, are a key part of managing and organising a project group, department or company.

Whilst technology trends may come and go, evolving rapidly to meet new requirements and market demands, there is also a complementary consideration of legacy, ongoing, and *tried and tested* solutions.

Production level systems may often last many years and cycles without significant modification and change. Such systems may endure for various reasons but they require consideration and attention as much as new and upcoming solutions and products.

For example, we might see long-term production systems for reasons such as

- slow release cycles for current production systems and technology stacks
  - e.g. due to requirements, features, logistics &c.
- company management may not wish to modify or potentially break their main revenue stream
  - i.e. if it's not broken, don't fix it…
- perception that the current production system cannot be improved or updated
  - a lack of incentive or motivation to improve or update
- management and technology support does not have appropriate experience beyond current technologies and stack
- …

Consideration of disaster recovery

Disaster recovery options, which, hopefully, will not need to be implemented, are still crucial for a successful, well considered project and company.

Commonly associated with initial large scale incidents, such as fire, flood, theft, and other potentially disastrous acts of vandalism or violence, disaster recoveryy will focus on backup solutions and provisions to transfer to a geographically difference location and operation.

Whilst this is easier with cloud native options, and is often provided as part of an overall service or product, it is still worthwhile to plan and prepare for disaster recovery relative to local work environments and services.

initial considerations As we begin to consider and define a potential plan for disaster recover, we might review the following considerations.

For example,

- what is permissible downtime for your app or service?
  - depending on the underlying business model and app, this might be seconds of a few days…
  - delays or downtime is rarely beneficial for client confidence or customer retention, so it should be as short as possible relative to underlying acceptible costs…
- does the app or service require specfic provision for restoration of state, data usage, &c.?
  - or is it able to reinstate a working service with a clean reset &c.?
  - such choices will, again, depend on the underlyin nature of the application, service, the business, its clients &c.
  - it may also be inferred by underlying costs
- as the app or service is being recovered, repaired, and prepared for restart, do admins &c. require full or partial access?
  - for example, akin to standard recovery options for various OSs, the admin user may only need access to underlying core systems, data validation tools &c. They may commonly not require direct access to user data, snapshots for persisted state, account details, and many other core functions for an app's default working environment…

Commonly, we end up with a review of these three foundational considerations and requirements for a disaster recovery plan.

- permitted downtime
- nature of recovery - partial or full
- recovery options - admin requirements

In addition to these initial, foundational considerations, we may also consider the nature of the backups we may user, and how and where we may recover the app or service. For example, whilst the point of failure might be a single data centre or provider, the recovery options may include sharded recovery centres in disparate locations. It's also useful to remember that once a recovery or backup location becomes active, due to issues and failures with the original primary location, the newly initiated backup will now require its own designated backup solution. In effect, provision for a cascade failure to ensure the *disaster recovery* plan is effective and viable beyond a single point of recovery.

permitted downtime Whilst it may sound preferable to define no downtime, or zero downtime to placate or please managers and executives, this will usually incur additional costs and resources beyond a standard backup and replace strategy.

Such options may also add further layers of complication to a disaster recovery plan, in particular for a company with disparate parts that do not fit well with the requirement for separate or multiple instances.

A common example of this issue is the setup and provision of databases for apps, clients, &c. For example, if a database is known to require minimal or zero downtime for company products, we might consider options to help ease potential offline usage, replication, and multi app usage. The use of a *Universally Unique Identifier* (UUID) or *Globally Unique Identifier* (GUID), helps provision resources beyond a standard database generated, auto-increment, primary key.

Whilst one option may initially be easier or preferable for a given database, if we then add the requirement for zero, or very low, downtime, we will need to consider the potential for sharding, and other forms of potential replication.

A UUID, or GUID, may generally be used to mitigate some of these issues, for example where a database or app will be defining an ID that must be different from an ID generated or defined by a potential third party beyond the admin or database's control. In the example of replication, a UUID may be used to ensure uniqueness.

However, even though UUIDs provide a high level of potential uniqueness, they do not inherently offer a guarantee of uniqueness. For such systems, including our disaster recovery plan, we may also need to define options to handle the rare case of a *collision* in the database and app's code.

synchronisation and backups As we introduce multiple, usually geographically diverse solutions, the underlying nature of backups will also need to consider effective synchronisation, reconciliation, and validation.

In effect, we will need to ensure data is kept in sync across multiple geographical areas, time zones &c. We will also need to consider the act of actually how and when to switch and migrate our users to another active zone.

Whilst physical constraints and performance continues to improve with network improvements, hardware updates, and cloud provisioned solutions, it still takes time to transfer and backup data from one location to another. Physical time zones, difference server locations may be a consideration for choice of backups in a company's recovery plan.

If we assume that the broader network connections remain active and viable for our current app, database, and service, we might still be limited by the time it takes to transfer data between disparate locations. Therefore, it's also important to review and consider network connections, or peer relationships, between chosen backup and recovery locations with the simple goal of trying to minimise the potential for general internet congestion and routing issues between data centres.

As with replication, databases are a useful example for a consideration of synchronisation, and its underlying limits or potential issues. For example, different database vendors, for example providing SQL variants, will provide different tiers for their products with variant levels of synchronisation and performance. Such offerings will, customarily, also incur higher costs for licensing, maintenance, backup, usage &c.

However, even with provisioned options for effective synchronisation, such database solutions may be limited by physical network resources and potential bottlenecks.

realistic solutions With these options in mind, and the potential for various limits and issues, it may simply not be practical or possible to offer an effective *zero* downtime solution as part of the broader *disaster recovery plan*.

As noted above, this may simply be due to network restrictions, performance constraints, or limited costs and funding.

For example, there are two common terms used to reference diverse solutions for a company's recovery plan. A *hot* and *cold* standby solution.

The *hot* solution may be prohibitively expensive, but it also assumes and, hopefully, delivers minimal downtime with servers continually in operation providing required synchronisation across disparate geographical zones. This solution should provide backup and recovery options to seamlessly transition from one solution to another, as required by a given, identified disaster. This option will also consume high levels of power for multiple data centre configurations, additional network bandwidth and provisions, and is consuming hardware resources in multiple, simulataneous regions. In effect, duplication costs across multiple zones.

By contrast, a *cold* solution requires a minimal number of machines and data centre provisions to ensure a system is able to recover for a given disaster. A minimum number of machines may be kept running for everyday operations, with new machines being added and started as required to solve a given problem or recover from a defined disaster. For example, a different geographical location may be initialised to start up machines, synchronise data, and begin providing an app or service. This may be the cheaper solution, but it will also required extra time for logistical and physical constraints. Thankfully, it is also usually the cheaper option for such recovery plans.

nature of recovery - partial or full For a given recovery plan, we commonly need to determine the extent and nature of the resumed service, including access permissions, network access, datastore or cached data, &c. In effect, should the recovered and resumed service be full or partial for the company and its clients.

A common consideration is the extent of the required service resumption without a full, detailed diagnostic and analysis. Perhaps, it will be possible to simply reboot and resume consumption of services. In other instances, it will be necessary to limit services, restrict access to live data, including new requests and updates, and mitigate the potential for exposure of data, accounts, &c.

Such reductions in service, such as a limit to posting new data, executing requests and transactions &c., are a common requirement for this type of plan . They provide a semblance of life and visibility for the application and service, whilst limiting the potential for further damage.

As the company's services and products are limited as part of a recovery plan, we need to clearly identify those places and parts that will become restricted and inaccessible to users. In addition to user facing components and places, we may also consider less visible limits we might impose on recovery system. For example, we might place restrictions on tools and services such as logging, user reporting, automated tests, any scheduled admin tasks, &c.

switch to full service As a system begins to migrate back to full, integrated service and application access, it is not necessarily a simple task of restarting or activating previously restricted places, services, and tools.

For example, it might become necessary to clearly audit systems for potential data loss, corruption of records, partial data writes, missing transactions, and so on. There are a myriad of checks that need to be performed to ensure balance is returned to a system and live application.

A related consideration is a reconciliation between the last known working state of the system, and the rebooted, live system. In effect, we might need to review and determine the last state of a system to enable a smooth transaction with the newly rebooted system. Will there be various transactions that require reprocessing for the new live system, or may we simply duplicate data from the old system to the new.

Regardless of the aims of the recovery plan, and its perceived best intentions, it may not be possible to provision a clean system with a seamless transition. As such, a decision may be required whether painstaking analysis and reconciliation is worth the additional costs relative to a clean reboot of the existing system.

fail forward As we implement a recovery plan, and potentially migrate service to a backup, geographically different location, we may need to consider and review what will happen when the original location resumes operations.

Do we simply return, or *bounce back*, to our previous setup and location? Or, do we consider updating our main site to the previously designated backup location?

In effect, if the backup location is now working at full capacity and performance with no discernible loss of perfomance, quality, or service, why incur the extra cost to simply return to the earlier setup? This scenario is commonly referenced as a *fail forward* approach to disaster recovery. You may then invert the previous relationship and now define the original location as the primary backup option.

However, if the transfer and backup is not a like for like transition, you may little option but to incur the cost and hassle of returning to the original location and server option.

making the switch The decision to activate a transfer from a primary setup or site to a designated backup is certainly open to many potential issues and errors. Therefore, a common consideration is simply when to activate a recovery plan. In effect, what is the tipping point to execute the plan, switch servers and locations, and begin informing users of the potential issues and consequences.

A disaster event, such as flood, fire, physical damage &c. may be simpler to define and identify, including when to activate the recovery plan itself. However, if the issue is more nuanced, or open to interpretation, such as power loss, network failure, hardware crashes &c., will this be sufficient justification to merit execution of the recovery plan.

As with other considerations in the recovery plan, this tipping point will need to be defined and clearly articulated. It might include loss of revenue, I/O inefficiency, time constraints, customer losses, and various other considerations. However, each company will need to identify their own balanced consideration of such limits.

As we cross this defined tipping point, we will then need clear directions and logistics for successfully executing and completing the switch to the recovery plan. For example, we might review the following options

- a defined order of precedence for service restarts, including the underlying components, data access, user accounts &c.
  - i.e. how may we begin the migration without introducing additional complications and issues
- are all passwords, codes, network config details, user details &c. current and accurate, and stored in a secure location
  - such resources will then need to be easily and readily accessible to the newly instantiated system
- which network details, including addresses, protocols, ports &c., need to be updated or migrated to the new systems and location
  - might include updates to DNS records for domain name, hosting, and underlying server reconciliation
  - such DNS updates, where necessary, may also require additional time to propagate
  - such times may also introduce vague estimates and potential limits on the speed of a execution for a recovery plan
- the identified backup location will also need to verified and tested prior to transfer and switch over from the original location
  - clearly identify trusted and appropriate roles, such as admin versus engineer, required to complete such analysis and preparatory tasks
  - who has the identified authority to confirm a backup and recovery location is ready for use
- who has the requisite authority to begin and execute the overall recovery plan
  - i.e. who makes the overall decision to implement a recovery plan for your services and applications…

The recovery plan should also be readily accessible by all personnel authorised to view and execute any of the underlying actions, as and when appropriate. This should also be replicated to avoid potential issues with the loss of the main server or service location.

check and update the plan The recovery plan itself should also be reviewed and checked on a regular basis.

For example, are data backups working as expected, are the personnel roles still valid, is the latency between live systems and recovery options still appropriate for the current application and compny demands, &c.

You might also consider regular tests and practice exercises or drills to ensure that each member of the recovery team is aware of their role and responsibility if the plan is executed in a real world scenario.

Resources

- “Design your organization to withstand future disasters” - Harvard Business Review - https://hbr.org/2022/03/design-your-organization-to-withstand-future-disasters
- “Designing disaster recovery strategies in a software product” - Rootstack - https://rootstack.com/en/blog/designing-disaster-recovery-strategies-software-product/
- “Disaster recovery plan examples (with how-to steps)” - Indeed - https://ca.indeed.com/career-advice/career-development/disaster-recovery-plan-example
- “Disaster recovery plans for IT ops and DevOps pros” - Atlassian - https://www.atlassian.com/incident-management/itsm/disaster-recovery
- “Gartner Research and Advice for Disaster Recovery” - Gartner - https://www.gartner.com/smarterwithgartner/gartner-research-and-advice-for-disaster-recovery
- Wallace, Michael and Webber, Lawrence. “The Disaster Recovery Handbook.” 3rd Ed. 2017. O’Reilly. - https://www.oreilly.com/library/view/the-disaster-recovery/9780814438770/

- “What is a disaster recovery team: roles for a successful plan” - Tierpoint - https://www.tierpoint.com/blog/disaster-recovery-team/