ECE590-02
Engineering Robust Server Software

Spring 2017

Business Continuity: Disaster Recovery

Tyler Bletsch
Duke University

Includes material adapted from the course "Information Storage and Management v2" (modules 9-12), published by EMC corporation.

Meta-notes

Notes I've added to the EMC stuff will appear in boxes like this one.

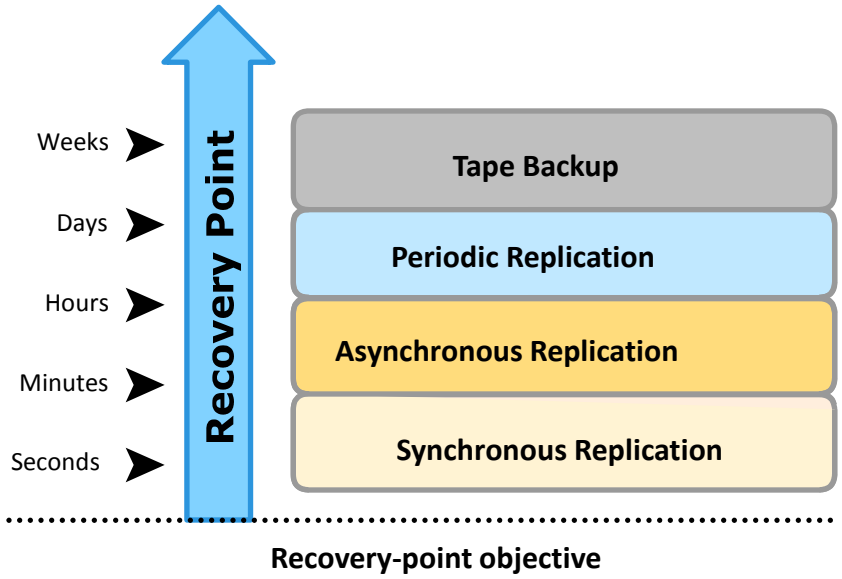
BC Terminologies – 1

- Disaster recovery
  - Coordinated process of restoring systems, data, and infrastructure required to support business operations after a disaster occurs
  - Restoring previous copy of data and applying logs to that copy to bring it to a known point of consistency
  - Generally implies use of backup technology
- Disaster restart
  - Process of restarting business operations with mirrored consistent copies of data and applications
  - Generally implies use of replication technologies

BC Terminologies – 2

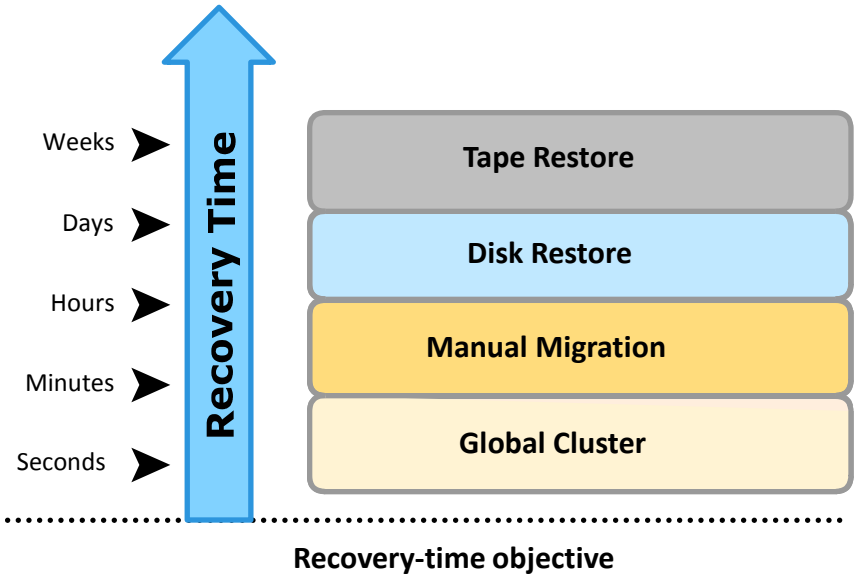
Recovery-Point Objective (RPO)

- Point-in-time to which systems and data must be recovered after an outage
- Amount of data loss that a business can endure



Recovery-Time Objective (RTO)

- Time within which systems and applications must be recovered after an outage
- Amount of downtime that a business can endure and survive



RPO vs RTO

Recovery Point Objective (RPO)

- How much did I lose?

Recovery Time Objective (RTO)

- How long until it's back?

Business Impact Analysis

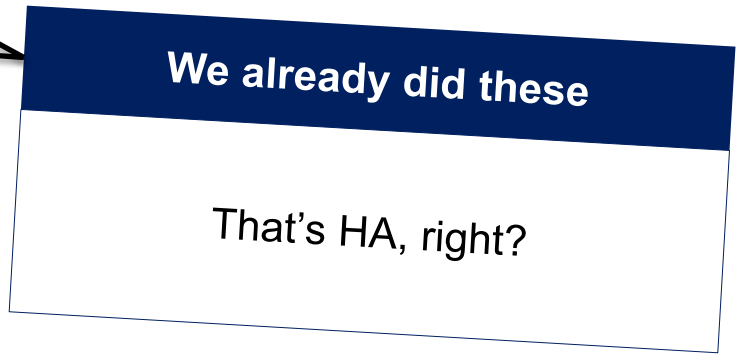
- Identifies which business units and processes are essential to the survival of the business
- Estimates the cost of failure for each business process
- Calculates the maximum tolerable outage and defines RTO for each business process
- Businesses can prioritize and implement countermeasures to mitigate the likelihood of such disruptions

Translation

Identify what will hurt the most to lose, spend your money there.

BC Technology Solutions

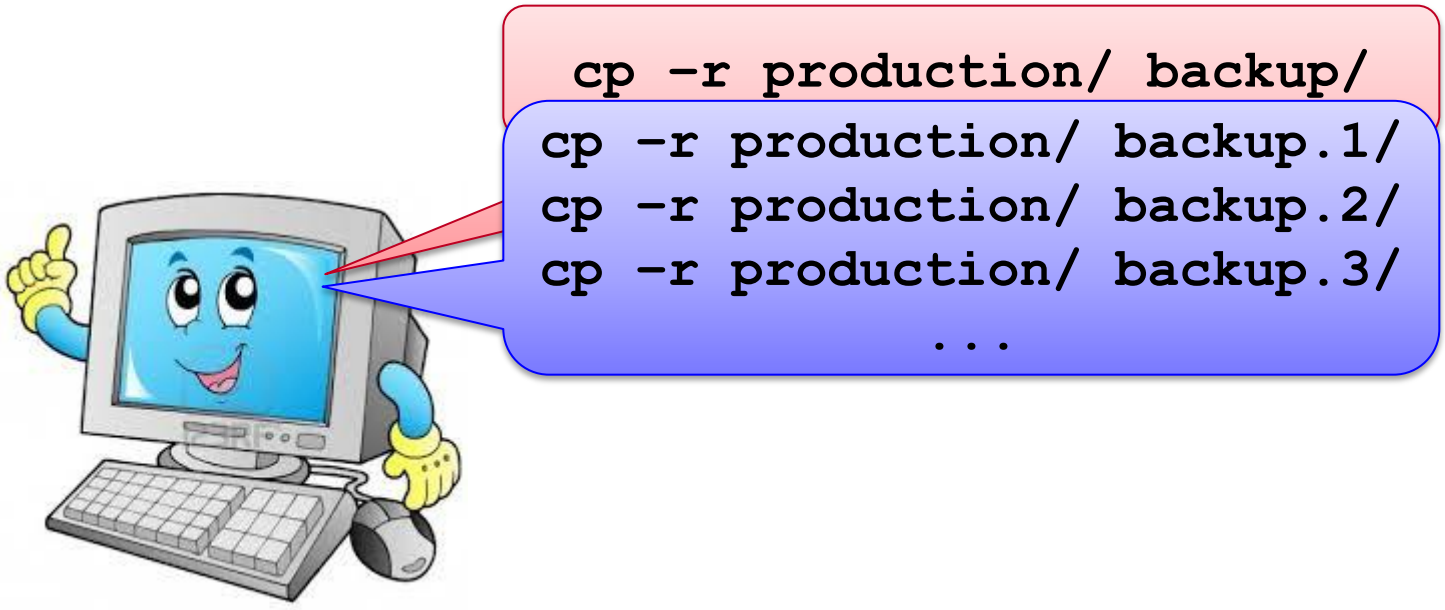
- Solutions that enable BC are:
  - Resolving single points of failure
  - Multipathing software
  - Backup and replication
    - Backup
    - Local replication
    - Remote replication



Continuous replication techniues omitted for time;
for details, take my Enterprise Storage Architecture ECE590

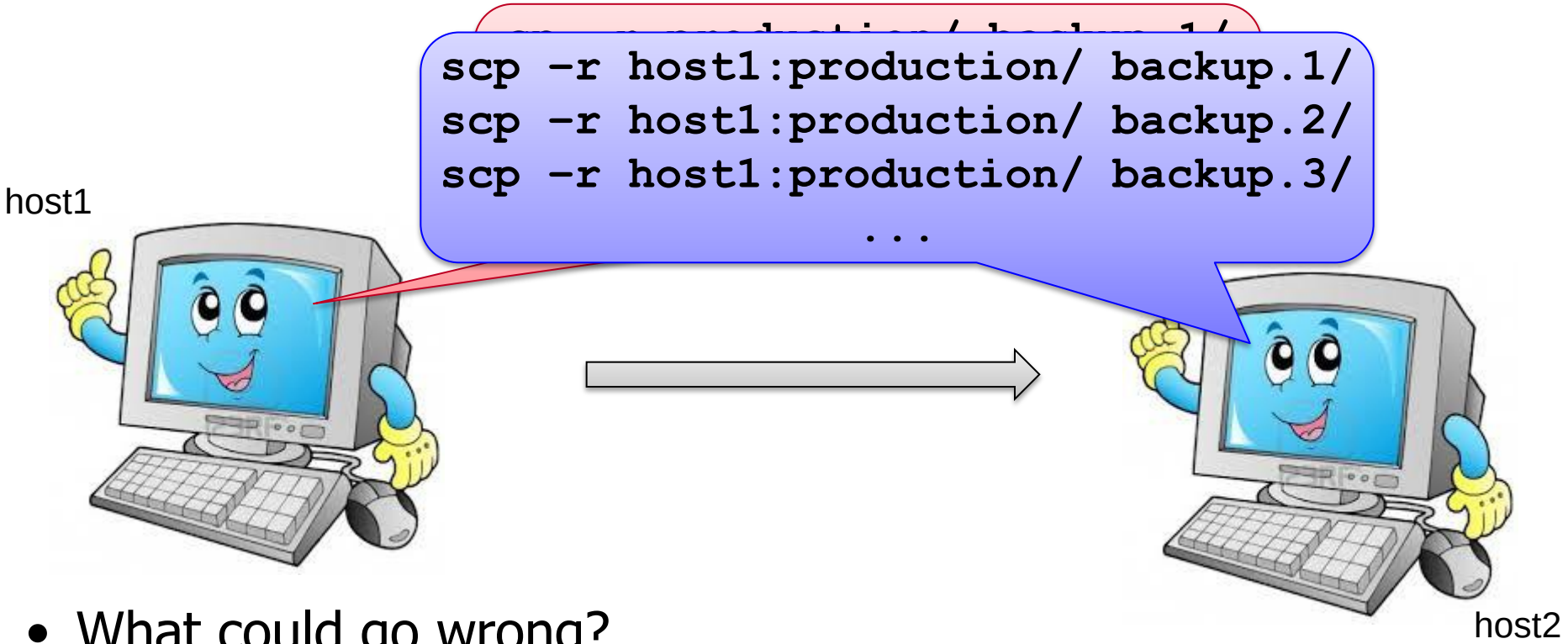
Backup and Archive

Building a backup solution



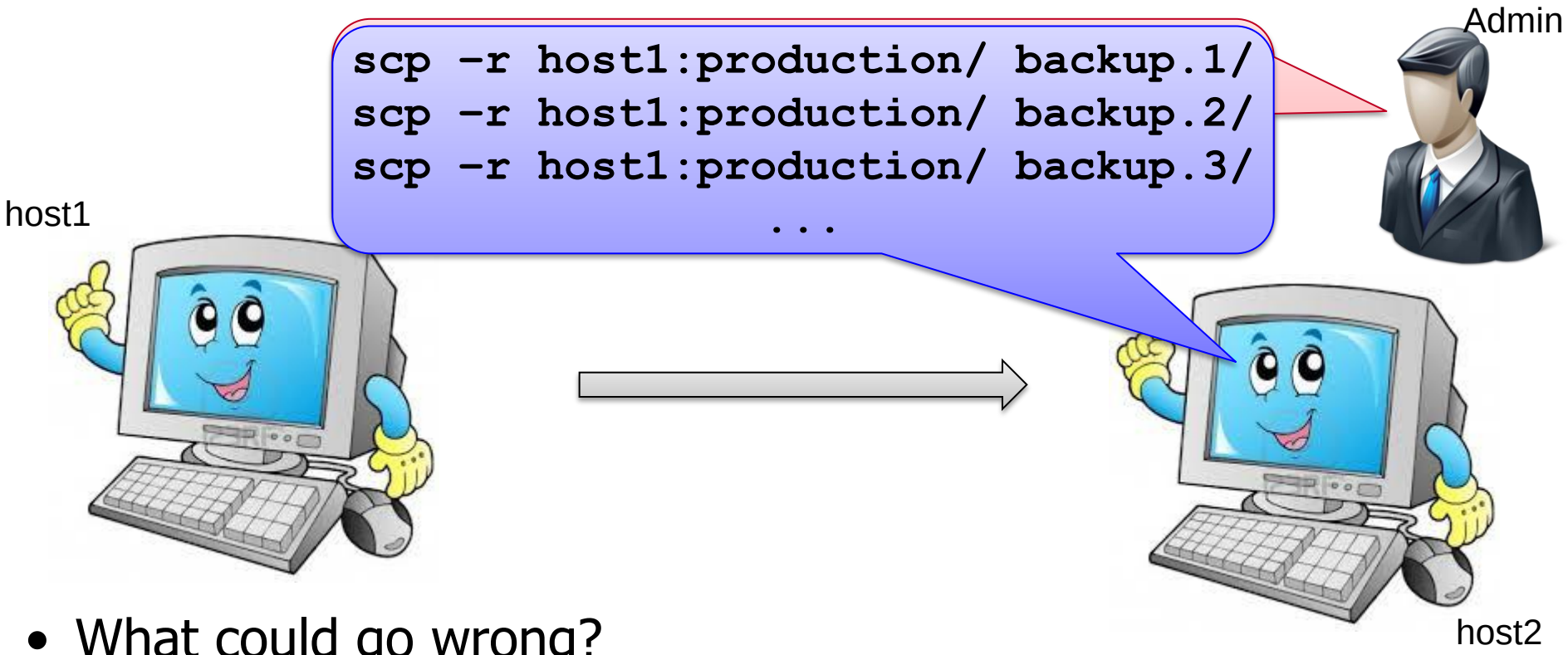
- What could go wrong?
  - Data corruption → Corrupt data overwrites backup → data loss
  - Solution: **multiple snapshots**

Building a backup solution



- What could go wrong?
  - System stolen/fails/destroyed → data loss
  - Solution: **separate systems**

Building a backup solution



- What could go wrong?
  - Admin forgets → data loss
  - Solution: **automation**

Building a backup solution

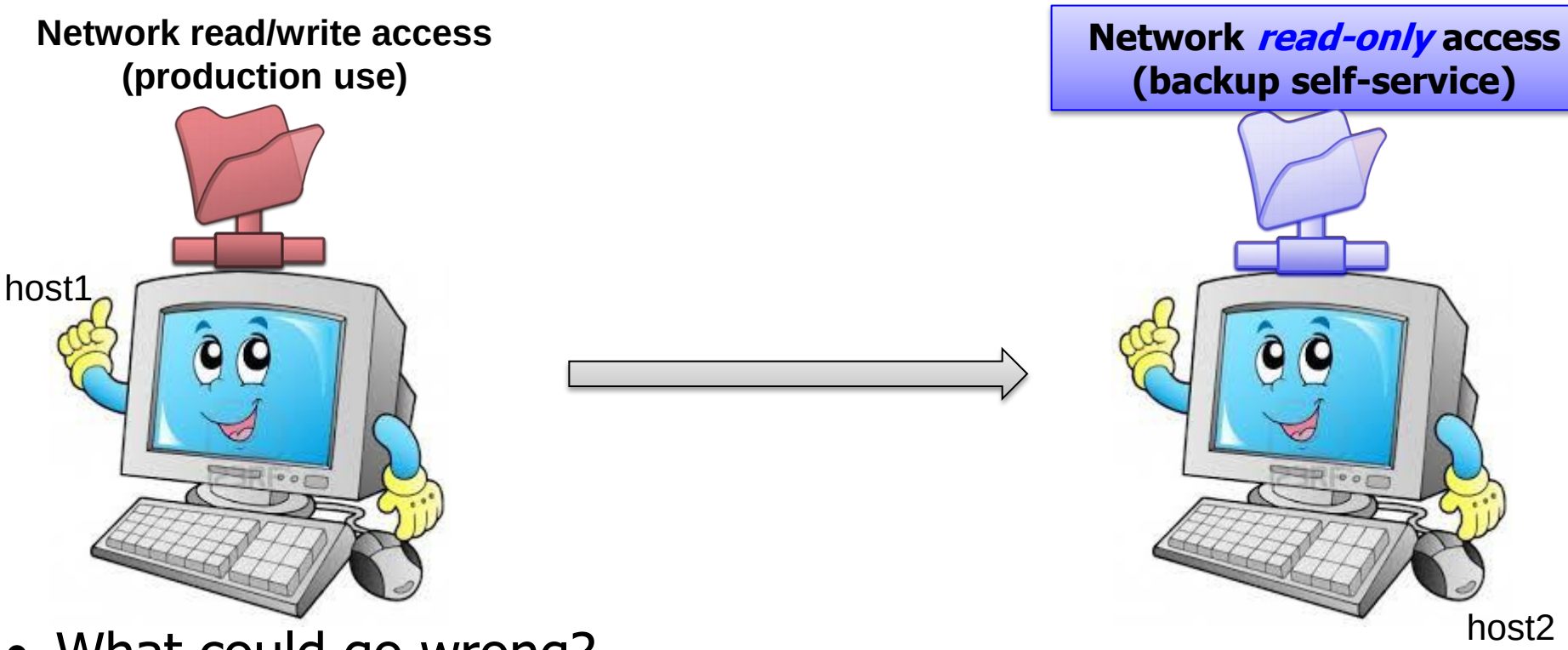
host1

Same admin access credentials

host2

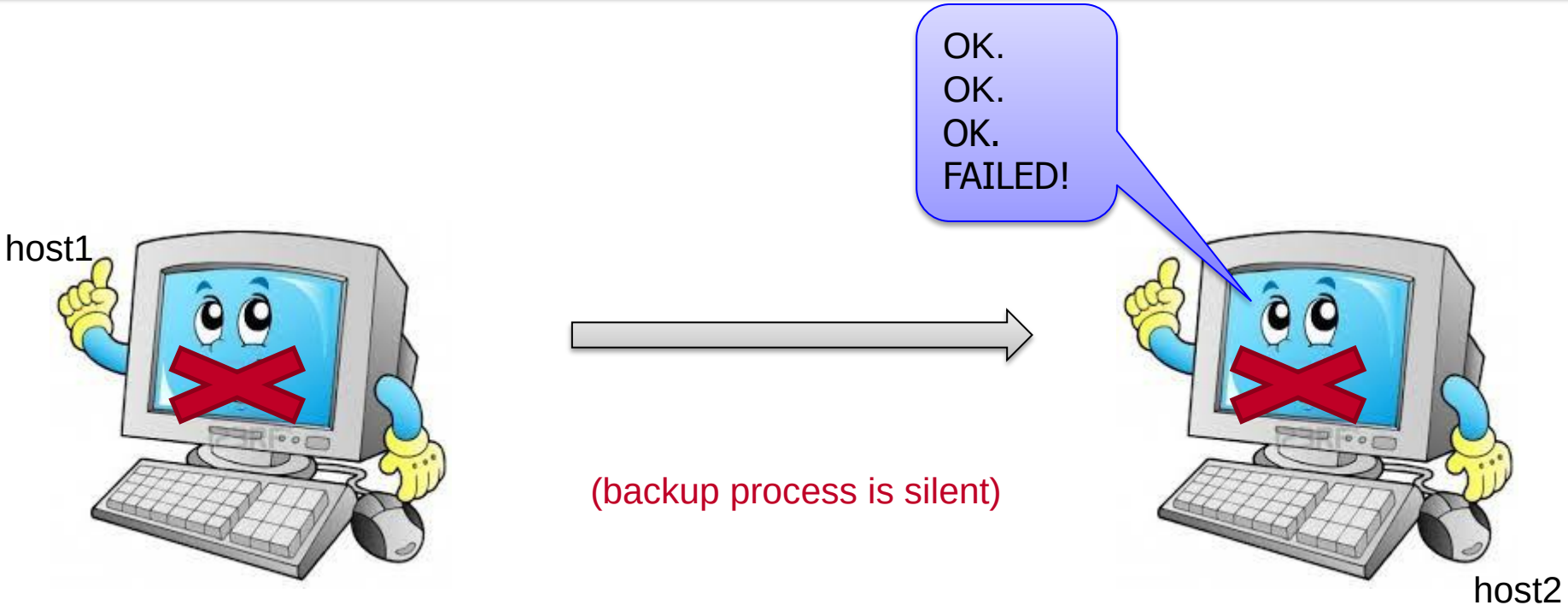
- What could go wrong?
  - Attacker gains one credential → Attacker can kill/corrupt all copies → data loss
  - Solution: **separate credentials for backup**

Building a backup solution



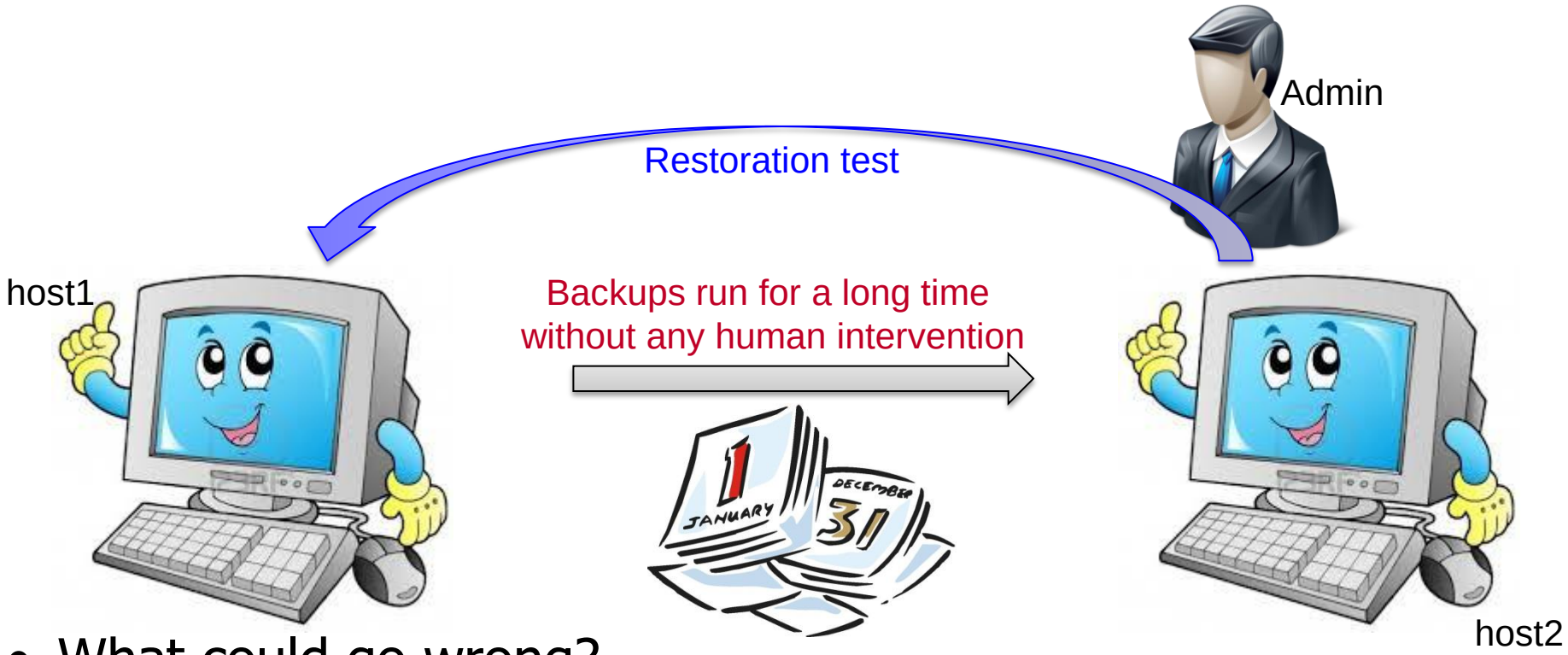
- What could go wrong?
  - User modifies primary and backup data → data loss
  - Solution: **backups must be unwritable**

Building a backup solution



- What could go wrong?
  - Backup server quietly dies → No backups kept for a while → data loss
  - Solution: **backups must report on success and alert on failure**

Building a backup solution



- What could go wrong?
  - Backups were done wrong all along → data not there when needed → data loss
  - Solution: **periodic restoration tests**

Tyler's Immutable Rules Of Backup
A BACKUP SOLUTION MUST:

1. Record changes to data **over time**
   - If I just have the most recent copy, then I just have the most recently corrupted copy. *RESULT: MIRRORING ISN'T BACKUP!!!!*
2. Have a copy at a **separate physical location**
   - If all copies are in one place, then a simple fire or lightning event can destroy all copies
3. Must be **automatic**
   - When you get busy, you'll forget, and busy people make the most important data
4. Require **separate credentials** to access
   - If one compromised account can wipe primary and secondary, then that account is a single point of failure
5. Be **unwritable** by anyone except the backup software (which ideally should live in the restricted backup environment)
   - If I can cd to a directory and change backups, then the same mistake/attack that killed the primary can kill the backup
6. Reliably **report** on progress and **alert** on failure
   - I need to know if it stopped working or is about to stop working
7. Have periodic **recovery tests** to ensure the right data is being captured
   - Prevent "well it apparently hasn't been backing up properly all along, so we're screwed"

If you encounter backups that don't meet these rules, explain the potential dangers until they do!

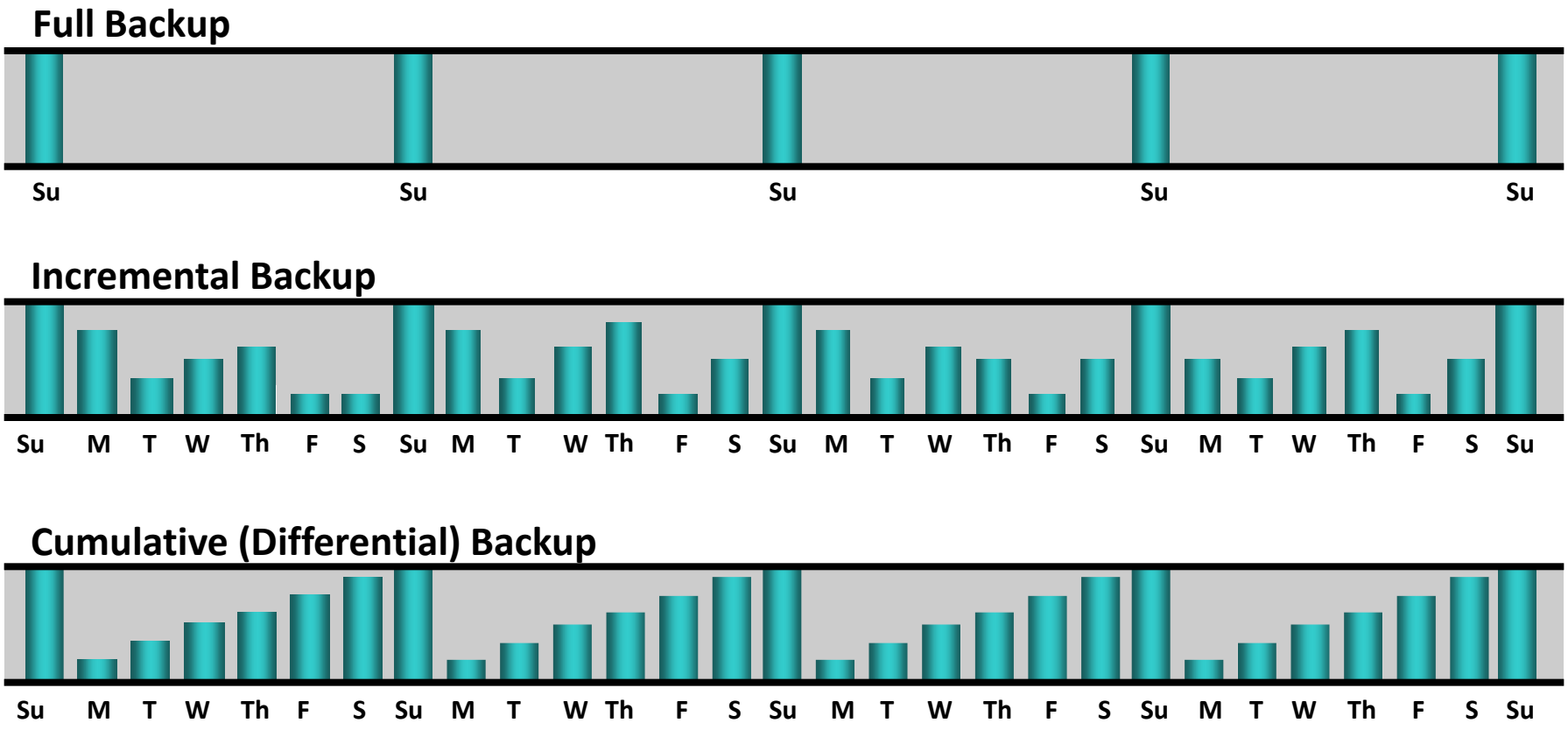
What is Backup?

Backup

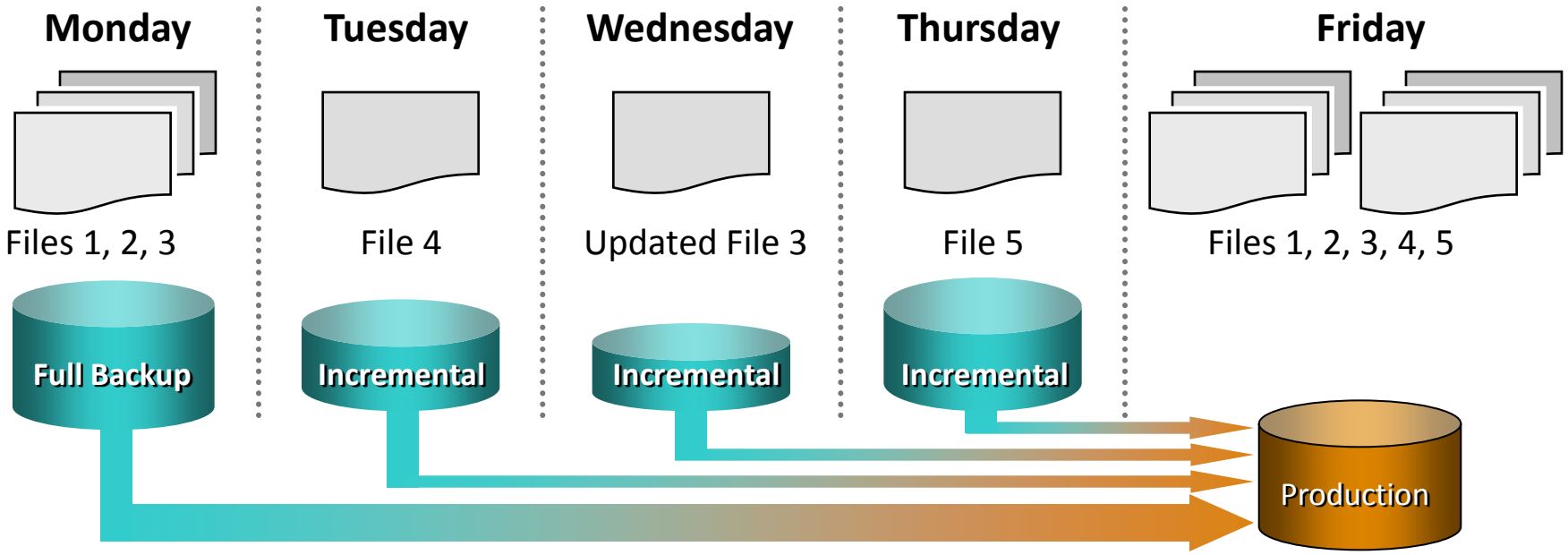
It is an additional copy of production data that is created and retained for the sole purpose of recovering lost or corrupted data.

- Organization also takes backup to comply with regulatory requirements
- Backups are performed to serve three purposes:
  - Disaster recovery
  - Operational recovery
  - Archive

Backup Granularity

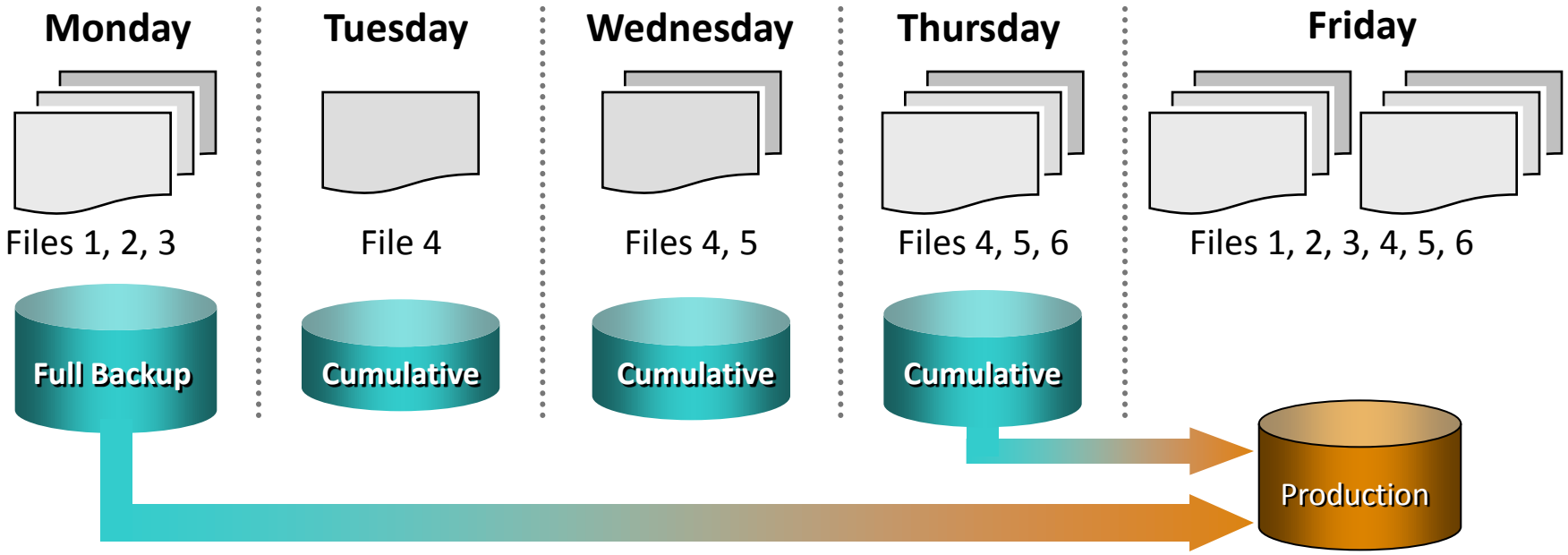


Restoring from Incremental Backup



- Less number of files to be backed up, therefore, it takes less time to backup and requires less storage space
- Longer restore because last full and all subsequent incremental backups must be applied

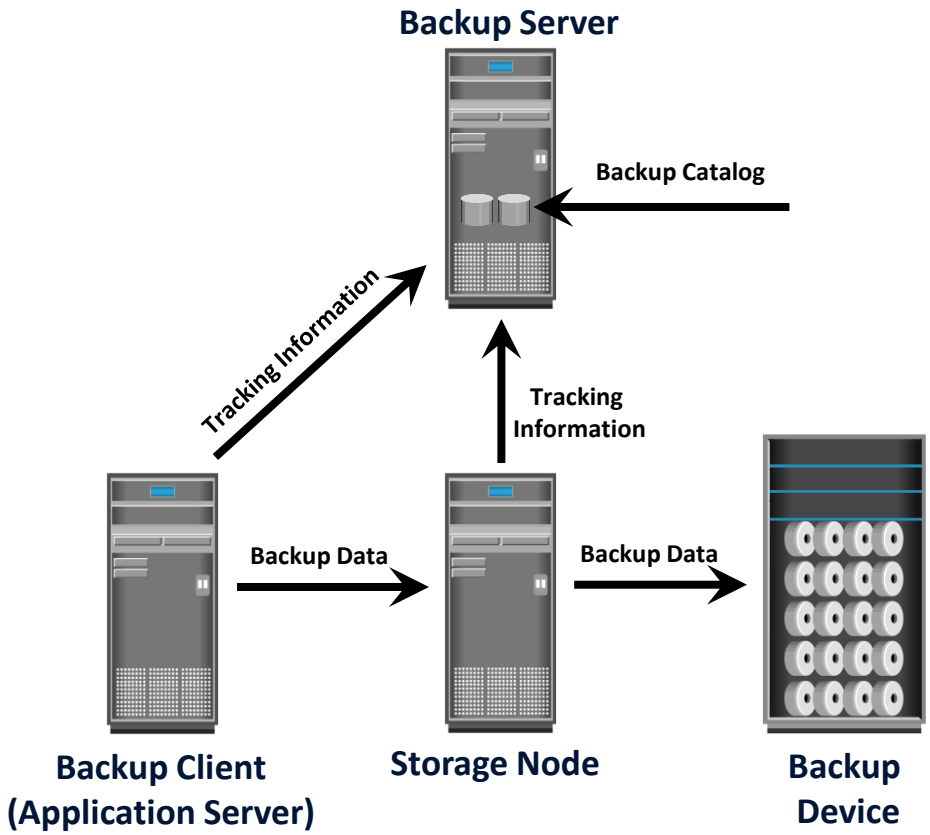
Restoring from Cumulative Backup



- More files to be backed up, therefore, it takes more time to backup and requires more storage space
- Faster restore because only the last full and the last cumulative backup must be applied

Backup Architecture

- Backup client
  - Gathers the data that is to be backed up and send it to storage node
- Backup server
  - Manages backup operations and maintains backup catalog
- Storage node
  - Responsible for writing data to backup device
  - Manages the backup device



I'm omitting a lot of discussion of how backup fits into enterprise-scale storage environments; for details, take my Enterprise Storage Architecture ECE590

Backup consistency

- Assume live ("hot") backup
- Is data crash-consistent, or can we do better?
- **Quiesce**: To make consistent at this time (**quiescent**).
  - Tell the OS that you're about to take a snapshot, request quiescence
  - OS flushes all buffers and commits the journal, pauses all IO, says OK
  - Take snapshot
  - Allow OS to resume
  - Base the backup (which takes longer) off this snapshot
  - Resulting backup is **OS consistent**
- Can also be application-aware
  - Same as above, but you tell the *application* to quiesce
  - Requires backup-aware applications (e.g. Microsoft SQL Server, Oracle database, etc.)
  - Resulting backups are **application consistent**

Backup targets

What media?

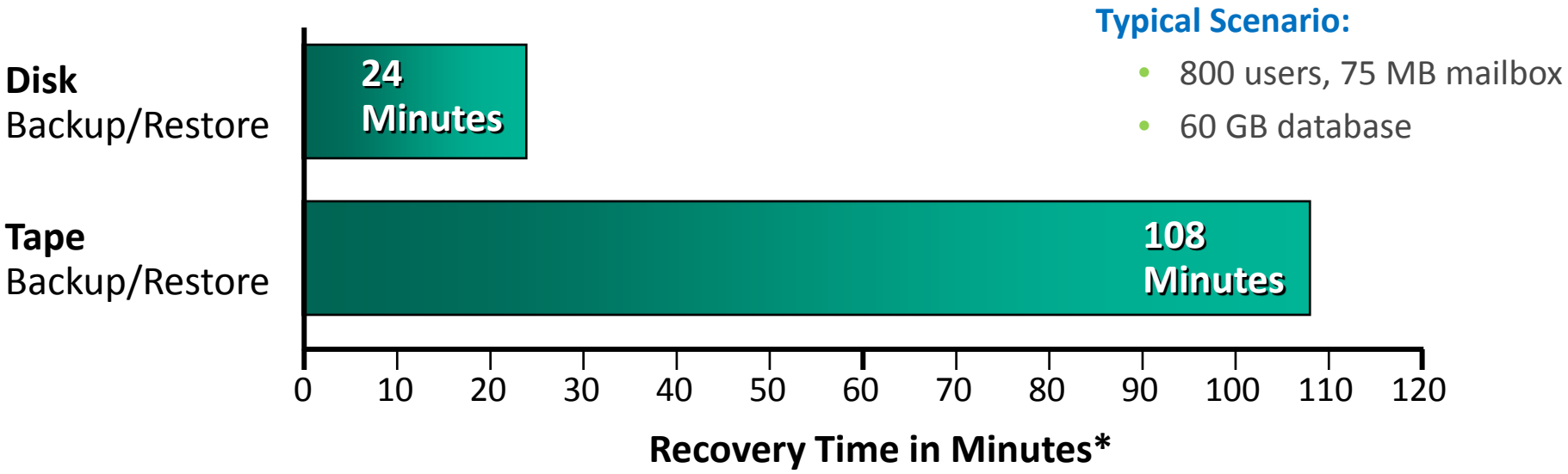
Where is it?

Backup to Tape

- Traditionally low cost solution
- Tape drives are used to read/write data from/to a tape
- Sequential/linear access
- Multiple streaming to improve media performance
  - Writes data from multiple streams on a single tape
- Limitation of tape
  - Backup and recovery operations are slow due to sequential access
  - Wear and tear of tape
  - Shipping/handling challenges
  - Controlled environment is required for tape storage
  - Causes “shoe shining effect” or “backhitching”

Backup to Disk

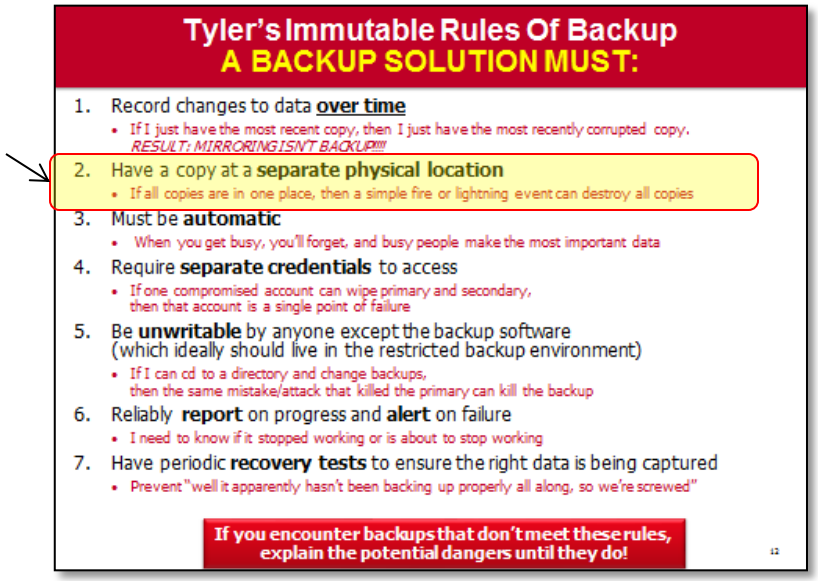
- Enhanced overall backup and recovery performance
  - Random access
- More reliable
- Can be accessed by multiple hosts simultaneously



I'm omitting exotic backup media, such as virtual tape, as well as the entire field of digital archival; for details, take my Enterprise Storage Architecture ECE590

Where does the backup go?

- **Local backup**
  - Use virtual "snapshots" or simple local media
  - But wait, don't my rules call for a separate physical location?
  - Yes, but local backup can be useful:
    - Can have lower RPO/RTO
    - Can be cheaper
    - May be sufficient for non-critical workloads where data loss is survivable



- Local backup is *useful* but not *sufficient* for business-critical workloads!

Where does the backup go?

- **Remote backup**
  - Use internet or dedicated link to go to different site
  - This is *not* HA (which has practical limits on distance, but we can switch to the remote site 'live'), but rather just backup (can *restore* from backup)
    - Result: no practical distance/latency limit
  - May have bandwidth limitations which limit rate-of-change of primary
    - If 5MB/s of new data is created, but backup link can do 4MB/s, then backup will get increasingly out-of-date over time
    - Eventually acts as bottleneck on primary!

Summary

- Disaster Recovery (DR) exists to handle cases where High Availability (HA) redundancy is overwhelmed
- For data, the key is backups;
  for compute, it's secondary compute servers
- Backup isn't just mirroring! Rules:
  1. Record changes to data **over time**
  2. Have a copy at a **separate physical location**
  3. Must be **automatic**
  4. Require **separate credentials** to access
  5. Be **unwritable** by anyone except the backup software
     (which ideally should live in the restricted backup environment)
  6. Reliably **report** on progress and **alert** on failure
  7. Have periodic **recovery tests** to ensure the right data is being captured
- Can do backup locally (for low cost, low RTO/RPO) and/or remotely (true DR, RTO/RPO proportional to cost)