Setting Up a Comprehensive Research Backup System on macOS

Research Backup Guide

Invalid Date

Table of contents

1 Introduction

Managing 300+ Git repositories across 20GB of research data requires a robust, automated backup strategy. This guide walks through setting up a three-tier backup system that provides Git-level versioning, real-time cloud sync, and comprehensive system backups.

1.1 Backup Strategy Overview

Our approach uses three complementary layers:

1. **Automated Git commits and pushes** (every 15 minutes)
2. **Cloud synchronization** (real-time via Google Drive/Dropbox)
3. **Time Machine backups** (hourly system-wide backups)

This ensures your research is protected against hardware failure, accidental deletion, Git corruption, and provides easy access across devices.

2 Setting Up Time Machine

Time Machine provides system-wide backup protection and serves as your safety net for everything beyond Git repositories.

2.1 Initial Time Machine Setup

2.1.1 Step 1: Connect Your USB Drive

1. Connect your 1TB USB drive to your MacBook
2. When prompted, **do not** use it for Time Machine yet - we'll configure this properly first

2.1.2 Step 2: Format the Drive (if needed)

1. Open **Disk Utility** (Applications > Utilities > Disk Utility)
2. Select your USB drive from the sidebar
3. Click **Erase**
4. Choose format: **Mac OS Extended (Journaled)** or **APFS** (recommended for newer Macs)
5. Name it something like "Research Backup"
6. Click **Erase**

2.1.3 Step 3: Configure Time Machine

1. Open **System Preferences** > **Time Machine**
2. Click **Select Backup Disk**
3. Choose your USB drive
4. Click **Use Disk**
5. If prompted about encryption, choose **Encrypt Backup** for security

2.1.4 Step 4: Customize Time Machine Settings

1. Click **Options** in Time Machine preferences
2. Add any folders you want to exclude (like Downloads, Trash, etc.)
3. **Important**: Do NOT exclude `~/prj` - we want this backed up
4. Ensure "Back up while on battery power" is enabled if desired

Time Machine will now automatically backup your entire system (including `~/prj`) every hour when the USB drive is connected.

3 Automated Git Backup Script

This script scans all Git repositories in `~/prj` every 15 minutes, commits changes, and pushes to GitHub.

3.1 Creating the Backup Script

3.1.1 Step 1: Create the Script File

Open Terminal and create the backup script:

```
mkdir -p ~/scripts
nano ~/scripts/backup-research.sh
```

3.1.2 Step 2: Add the Script Content

Copy and paste this script:

```
#!/bin/bash

# Research Git Backup Script
# Automatically commits and pushes changes in all Git repositories

RESEARCH_DIR="$HOME/prj"
LOG_FILE="$HOME/Library/Logs/research-backup.log"
MAX_LOG_SIZE=10485760  # 10MB

# Create log directory if it doesn't exist
mkdir -p "$(dirname "$LOG_FILE")"

# Rotate log if it gets too large
if [[ -f "$LOG_FILE" && $(stat -f%z "$LOG_FILE") -gt $MAX_LOG_SIZE ]]; then
    mv "$LOG_FILE" "${LOG_FILE}.old"
fi

# Function to log messages
log_message() {
    echo "$(date '+%Y-%m-%d %H:%M:%S'): $1" >> "$LOG_FILE"
}

log_message "Starting research backup scan"

# Check if research directory exists
if [[ ! -d "$RESEARCH_DIR" ]]; then
    log_message "ERROR: Research directory $RESEARCH_DIR does not exist"
    exit 1
```

```
fi

# Counter for repositories processed
repo_count=0
backup_count=0

# Find all .git directories and process them
find "$RESEARCH_DIR" -name ".git" -type d | while read -r git_dir; do
    repo_dir=$(dirname "$git_dir")
    repo_name=$(basename "$repo_dir")

    cd "$repo_dir" || {
        log_message "ERROR: Cannot access $repo_dir"
        continue
    }

    repo_count=$((repo_count + 1))

    # Check if there are uncommitted changes
    if [[ -n $(git status --porcelain) ]]; then
        # Stage all changes
        git add -A

        # Create commit with timestamp
        commit_message="Auto-backup: $(date '+%Y-%m-%d %H:%M:%S')"

        if git commit -m "$commit_message"; then
            # Try to push to remote
            # First try 'main' branch, then 'master'
            if git push origin main 2>/dev/null || git push origin master 2>/dev/null; then
                log_message "SUCCESS: Backed up and pushed $repo_name"
                backup_count=$((backup_count + 1))
            else
                log_message "WARNING: Committed $repo_name but failed to push (no remote or
            fi
        else
            log_message "ERROR: Failed to commit changes in $repo_name"
        fi
    fi
done

log_message "Backup scan complete. Processed $repo_count repositories, backed up $backup_cou
```

3.1.3 Step 3: Make the Script Executable

```
chmod +x ~/scripts/backup-research.sh
```

3.1.4 Step 4: Test the Script

Run it once manually to ensure it works:

```
~/scripts/backup-research.sh
```

Check the log file to see results:

```
tail -20 ~/Library/Logs/research-backup.log
```

3.2 Setting Up Automated Execution

3.2.1 Step 1: Create the Cron Job

Open your crontab for editing:

```
crontab -e
```

Add this line to run the script every 15 minutes:

```
# Research backup - runs every 15 minutes
*/15 * * * * /Users/$(whoami)/scripts/backup-research.sh
```

3.2.2 Step 2: Verify Cron Job

List your cron jobs to confirm:

```
crontab -l
```

3.3 Monitoring the Backup System

3.3.1 View Recent Backup Activity

```
tail -50 ~/Library/Logs/research-backup.log
```

3.3.2 Check for Errors

```
grep "ERROR\|WARNING" ~/Library/Logs/research-backup.log
```

3.3.3 Monitor in Real-Time

```
tail -f ~/Library/Logs/research-backup.log
```

4 Cloud Synchronization Setup

Adding cloud sync provides real-time backup and cross-device access to your research files.

4.1 Recommended: Google Drive Setup

1. **Install Google Drive for Desktop** from drive.google.com
2. **Sign in** with your Google account
3. **Configure sync location**:
    - Choose "Mirror files" option
    - Select a location like `~/GoogleDrive`
4. **Move your research directory**:

    ```
    # Create backup first
    cp -r ~/prj ~/prj-backup

    # Move to Google Drive
    mv ~/prj ~/GoogleDrive/prj

    # Create symlink at original location
    ln -s ~/GoogleDrive/prj ~/prj
    ```

4.2 Alternative: Dropbox Setup

1. **Install Dropbox** from dropbox.com
2. **Sign in** and complete setup
3. **Move research directory**:

   ```
   # Create backup first
   cp -r ~/prj ~/prj-backup

   # Move to Dropbox
   mv ~/prj ~/Dropbox/prj

   # Create symlink
   ln -s ~/Dropbox/prj ~/prj
   ```

The symlink ensures your backup script continues working with the original `~/prj` path while files are actually stored in the cloud service folder.

5 Backup System Verification

5.1 Daily Checks

5.1.1 1. Verify Time Machine Status

```
tmutil status
```

5.1.2 2. Check Recent Git Backups

```
tail -20 ~/Library/Logs/research-backup.log | grep "SUCCESS"
```

5.1.3 3. Confirm Cloud Sync

Check that recent changes appear in your cloud service's web interface.

5.2 Weekly Health Check

5.2.1 1. Test Repository Recovery

Pick a test repository and verify you can: - See recent auto-commits in Git history - Access files through cloud service web interface - Restore from Time Machine if needed

5.2.2 2. Check Backup Coverage

```
# Count total repositories
find ~/prj -name ".git" -type d | wc -l

# Check log for recent activity
grep "$(date '+%Y-%m-%d')" ~/Library/Logs/research-backup.log | wc -l
```

6 Troubleshooting

6.1 Common Issues and Solutions

6.1.1 Script Not Running Automatically

Problem: Cron job isn't executing the script

Solutions: 1. Check cron is running: `sudo launchctl list | grep cron` 2. Verify script permissions: `ls -la ~/scripts/backup-research.sh` 3. Check for syntax errors in crontab: `crontab -l`

6.1.2 Git Push Failures

Problem: Repositories aren't pushing to GitHub

Solutions: 1. Verify SSH keys or credentials are configured 2. Check repository remotes: `git remote -v` 3. Test manual push in a repository

6.1.3 Time Machine Not Backing Up

Problem: Time Machine shows errors or isn't running

Solutions: 1. Check disk space on backup drive 2. Verify drive is properly connected and mounted 3. Run First Aid on backup drive in Disk Utility

6.1.4 Cloud Sync Issues

Problem: Files not syncing to cloud service

Solutions: 1. Check internet connection 2. Verify cloud service client is running 3. Check for file conflicts in cloud service interface

7 Conclusion

This three-tier backup system provides comprehensive protection for your research:

- **15-minute Git automation** ensures no work is lost and maintains proper version control
- **Real-time cloud sync** provides immediate off-site backup and device accessibility

- **Hourly Time Machine backups** protect against system failures and provide easy file recovery

The system runs automatically once configured, requiring minimal maintenance while providing maximum protection for your valuable research data.

7.1 Maintenance Schedule

- **Daily**: Quick log check for any error messages
- **Weekly**: Verify all three backup layers are functioning
- **Monthly**: Review and clean up old log files
- **Quarterly**: Test full recovery process with a sample repository

Your research is now protected against virtually any data loss scenario!