🚀 Quick Version Control & Backup Demo

This guide shows you how to use the version control and backup system step-by-step.

☑ Current Status

Your system is now version controlled with Git!

- **Repository:** Initialized ☑
- **Initial Version:** v3.0.0 ☑
- **Backup Scripts:** Ready ☑

📋 Quick Commands

1. View Current Version

```
git describe --tags
# Output: v3.0.0
```

2. View Version History

```
git log --oneline --decorate
# Shows all commits with tags
```

3. List All Versions

```
git tag -l
# Lists all version tags
```

🏷 How to Create a New Version

Example: After fixing a bug

Step 1: Make your changes

```
# Edit files, make improvements, fix bugs
```

Step 2: Check what changed

```
git status
git diff
```

Step 3: Commit changes

```
git add .
git commit -m "Fixed HOD approval bug where comments were not saving"
```

Step 4: Create version tag

```
git tag -a v3.0.1 -m "Version 3.0.1 - Bug fix: HOD approval comments"
```

Step 5: Verify

```
git tag -l
# Should show: v3.0.0, v3.0.1
```

Using the Script (Easier Method)

```
# Windows Command Prompt or PowerShell
cd C:\projects\purchase-requisition-system
scripts\create-version.bat v3.0.1 "Bug fix: HOD approval"
```

The script will:

- Ask if you want to commit uncommitted changes
- Create the commit
- Create the git tag
- Show confirmation

💾 How to Backup Each Version

Method 1: Manual Backup (Using Windows Explorer)

Step 1: Create backup folder

```
C:\Backups\PurchaseRequisitionSystem\
```

Step 2: Copy project folder

- Copy entire `C:\projects\purchase-requisition-system` folder
- Rename to: `PRS-v3.0.0-backup-2025-01-06`
- Compress to ZIP for storage

Method 2: Using Backup Script

Windows Command Prompt:

```
cd C:\projects\purchase-requisition-system
scripts\backup-version.bat v3.0.0
```

What it creates:

```
C:\Projects\purchase-requisition-backups\
  ├── prs-v3.0.0-20250106_143022\         (Folder)
  └── prs-v3.0.0-20250106_143022.zip      (Compressed)
```

Method 3: Git Archive (Quick Code-Only Backup)

Export specific version:

```
git archive --format=zip --output=backup-v3.0.0.zip v3.0.0
```

This creates a ZIP file with just the code (no database or uploads).

Complete Workflow Example

Scenario: Adding a New Feature

1. Current state

```
git status
# On branch master, v3.0.0
```

2. Create backup before making changes

```
# Option A: Manual copy
# Copy folder and rename: PRS-backup-before-email-feature

# Option B: Git tag (no changes yet, so we're at v3.0.0)
git describe --tags
# v3.0.0
```

3. Make changes

```
# Edit files to add email notification feature
# Test thoroughly
```

4. Commit changes

```
git add .
git commit -m "Feature: Add email notifications for approval stages

- Added email service integration
- Notifications sent on approval/rejection
- Configurable email templates
- Tested with Gmail SMTP"
```

5. Create new version

```
git tag -a v3.1.0 -m "Version 3.1.0 - Email Notifications Feature"
```

6. Backup the new version

```
scripts\backup-version.bat v3.1.0
```

7. Verify everything

```
git tag -l
# v3.0.0
# v3.1.0

git log --oneline
# Shows both commits
```

Backup Types Explained

Type 1: Code-Only Backup (Git Archive)

```
git archive --format=zip --output=backup-code-v3.0.0.zip v3.0.0
```

Includes:

- ☑ All source code
- ☑ Configuration files
- ☑ Documentation

Excludes:

- ❌ Database files
- ❌ Uploaded files
- ❌ .env file

- ❌ node_modules

Use when: Sharing code, storing in version control

Size: ~1-5 MB

Type 2: Full System Backup (Backup Script)

```
scripts\backup-version.bat v3.0.0
```

Includes:

- ☑ All source code
- ☑ Database files (.db)
- ☑ Uploaded files
- ☑ .env configuration
- ☑ Documentation

Excludes:

- ❌ node_modules (can be reinstalled)
- ❌ Log files

Use when: Before major changes, regular backups

Size: ~10-50 MB (depends on uploads)

Type 3: Complete Clone (Manual Copy)

Includes:

- ☑ Everything including node_modules

Use when: Emergency backup, moving to new machine

Size: ~200-500 MB

Practical Examples

Example 1: Bug Fix Workflow

```
# 1. Notice bug in HOD approval
# 2. Create backup
git tag -a v3.0.0-pre-fix -m "Before HOD fix"

# 3. Fix the bug
# Edit backend/routes/approvals.js

# 4. Test fix
npm test

# 5. Commit
git add backend/routes/approvals.js
git commit -m "Fix: HOD approval comments not saving to database"

# 6. Create patch version
git tag -a v3.0.1 -m "Version 3.0.1 - Bug fix"

# 7. Backup
scripts\backup-version.bat v3.0.1
```

Example 2: Feature Development

```
# 1. Starting new feature
git checkout -b feature-email-notifications

# 2. Develop feature
# ... make changes ...

# 3. Test feature
npm test

# 4. Merge to master
git checkout master
git merge feature-email-notifications

# 5. Create minor version
git tag -a v3.1.0 -m "Version 3.1.0 - Email notifications"

# 6. Backup
scripts\backup-version.bat v3.1.0

# 7. Delete feature branch
git branch -d feature-email-notifications
```

Example 3: Emergency Rollback

```
# Something went wrong!
# Need to go back to v3.0.0

# Option A: Restore from backup
scripts\restore-version.bat prs-v3.0.0-20250106_143022

# Option B: Git checkout
git checkout v3.0.0

# Option C: Reset to tag
git reset --hard v3.0.0
```

Recommended Backup Schedule

Daily (Automated)

- **What:** Current state backup
- **When:** 2:00 AM
- **Keep:** 30 days
- **How:** Use Task Scheduler + automated-backup.bat

Weekly (Manual)

- **What:** Tagged version backup
- **When:** Friday EOD
- **Keep:** 90 days
- **How:** `scripts\backup-version.bat v3.0.x`

Monthly (Manual)

- **What:** Full archive backup
- **When:** Last day of month
- **Keep:** Permanently
- **How:** Copy to external drive

Before Changes (Manual)

- **What:** Safety backup
- **When:** Before any major update
- **Keep:** Until verified
- **How:** Manual copy or git tag

How to Find a Specific Version

Find by Date

```
git log --since="2025-01-01" --until="2025-01-07" --oneline
```

Find by Message

```
git log --grep="email" --oneline
```

Find by File

```
git log --follow -- backend/server.js
```

Find by Author

```
git log --author="John" --oneline
```

Find Tag by Name

```
git tag -l "v3.0.*"
# Shows: v3.0.0, v3.0.1, v3.0.2
```

Verification Checklist

After creating a version and backup:

- [ ] Version tag created: `git tag -l`
- [ ] Commit has good message: `git log -1`
- [ ] Backup file exists: Check backup directory
- [ ] Backup size is reasonable: 10-50 MB
- [ ] Backup contains database: Check .zip contents
- [ ] Application still works: Test it!
- [ ] Document version: Update CHANGELOG.md

Troubleshooting

"Not a git repository"

```
cd C:\projects\purchase-requisition-system
git status
# Should show current status
```

"Tag already exists"

```
# Delete old tag
git tag -d v3.0.1
# Create new tag
git tag -a v3.0.1 -m "New message"
```

"Nothing to commit"

```
# Check status
git status
# This is normal if no changes made
```

Backup script doesn't run

```
# Run from Command Prompt, not Git Bash
# Make sure you're in project root directory
cd C:\projects\purchase-requisition-system
```

Quick Help

Problem	Solution
Forgot version number	`git describe --tags`
Want to undo last commit	`git reset HEAD~1`
Want to see changes	`git diff`
Want to see history	`git log --oneline --graph`
Need to restore	`scripts\restore-version.bat backup-name`
Backup failed	Check disk space, run as administrator

Learn More

- Full guide: `VERSION_CONTROL_GUIDE.md`
- Git basics: https://git-scm.com/doc
- Troubleshooting: `TROUBLESHOOTING.md`

Ready to start?

1. Check current version: `git describe --tags`
2. Make a change to test
3. Commit it: `git add . && git commit -m "Test commit"`
4. Create tag: `git tag -a v3.0.1-test -m "Test version"`
5. View it: `git log --oneline --graph`

You're all set! 🎉