

Automates Git workflow for deploying changes to remote repositories. Handles the standard addcommit-push cycle with error checking and user feedback.

Purpose

- Automated Deployment: Reduces manual Git commands
- Error Handling: Provides clear feedback on failures
- GitHub Pages Integration: Optimized for static site deployment
- Zero-downtime: Checks for changes before committing

Script Breakdown

1. Configuration Variables

```
bash

GIT_BRANCH="main"

COMMIT_MESSAGE="Updated content"
```

Customizable Settings:

- Branch target: Usually "main" or "master"
- Default message: Generic but descriptive
- Easy modification: Change once, applies to all commits

2. Git Add Process

```
bash

echo "Adding all changes to Git staging area..."
glt add .
```

What git add . does:

- Stages all changes in current directory and subdirectories
- Includes new files: Newly generated PNGs and updated HTML
- Includes modifications: Changed USDZ files or updated content
- Includes deletions: Removed files are staged for deletion

3. Change Detection

```
bash

if git diff --cached --quiet; then
echo "No changes to commit. Exiting."
exit 0
fi
```

Smart Skip Logic:

- (git diff --cached): Shows staged changes
- (--quiet) flag: No output, just exit code
- Exit code 0: No differences found
- Early exit: Prevents empty commits and unnecessary pushes

Why this matters:

- Saves time on repeated runs
- · Prevents cluttering Git history with empty commits
- Provides clear feedback when nothing needs deployment

4. Commit Process

```
echo "Committing changes with message: \"$COMMIT_MESSAGE\""
git commit -m "$COMMIT_MESSAGE"
```

Commit Creation:

- Fixed message: Simple but identifiable
- All staged changes: Includes thumbnails, HTML, and any other updates
- Local operation: Fast, doesn't require network

5. Remote Push

```
bash

echo "Pushing changes to $GIT_BRANCH branch on GitHub..."

git push origin "$GIT_BRANCH"
```

Push Process:

- Remote sync: Sends local commits to GitHub
- Branch specific: Pushes to configured branch only
- Triggers deployment: GitHub Pages rebuilds site automatically

6. Exit Status Handling

```
bash

if [ $? -eq 0 ]; then
    echo "--- Git Deployment Successful! ---"
    echo "Your changes should now be deploying to your live site."

else
    echo "--- Git Deployment Failed! ---"
    echo "Please check the error messages above..."

fi
```

Error Reporting:

- (\$?): Exit code of last command (git push)
- Exit code 0: Success
- Non-zero: Failure (authentication, conflicts, network)
- User guidance: Suggests common solutions

Common Issues and Solutions

generate_index_with_USDZ.sh Issues

No USDZ files found:

- Script handles gracefully with (2>/dev/null)
- · Creates empty gallery page
- No error messages

Filenames with special characters:

- IFS handling manages spaces correctly
- URL encoding prevents link breakage
- HTML escaping prevents injection

Missing thumbnails:

- Graceful fallback to memoji placeholder
- · Consistent layout maintained
- · No broken image links

deploy.sh Issues

Authentication failures:

- Usually requires Personal Access Token
- Check GitHub authentication settings
- · Verify repository permissions

Merge conflicts:

- Requires (git pull) before push
- · Script doesn't handle this automatically
- Manual intervention needed

Network issues:

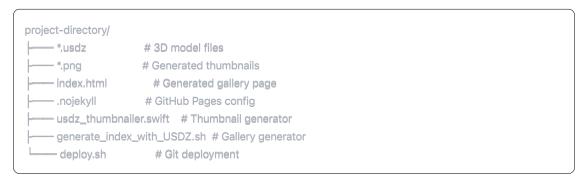
- · Retry the deployment
- Check internet connection
- · Verify GitHub status

Integration Workflow

Complete Process Flow

- 1. Edit USDZ files (add/modify/remove models)
- 2. Run USDZ thumbnailer (generates PNGs)
- 3. **generate_index_with_USDZ.sh** (creates gallery HTML)
- 4. deploy.sh (pushes to GitHub Pages)
- 5. GitHub Pages rebuilds (site updates automatically)

File Dependencies



Typical Output

```
bash
# From thumbnailer

√ model1.usdz

✓ model2.usdz

- model3.usdz (up to date)
Done: 2 success, 0 failed
# From index generator
--- Generating index.html and .nojekyll file ---
.nojekyll file ensured for GitHub Pages.
index.html has been generated with links to all .usdz files.
--- Generation Complete ---
# From deployment
--- Starting Git Deployment ---
Adding all changes to Git staging area...
Committing changes with message: "Updated content"
Pushing changes to main branch on GitHub...
--- Git Deployment Successful! ---
Your changes should now be deploying to your live site.
```

Customization Options

generate_index_with_USDZ.sh

- Styling: Modify CSS in the (<style>) section
- Colors: Change Tailwind classes (bg-blue-600, etc.)
- Layout: Adjust file-item structure
- Branding: Update header text and footer

deploy.sh

- Branch: Change (GIT_BRANCH) variable
- Commit message: Modify (COMMIT_MESSAGE)
- Pre-push checks: Add additional validation
- Post-deploy actions: Add notification webhooks