

# Git Stash — Commands, Explanations & Examples

This concise guide explains common git stash commands with practical examples and scenarios. Use these commands to temporarily save uncommitted changes, switch branches safely, or create focused work branches from stashed changes.

## What is git stash?

`git stash` temporarily shelves (or "stashes") changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on.

## Common workflows

- 1) You are in the middle of work but need to switch branches.
- 2) You want to keep uncommitted changes but create a new branch from them.
- 3) You want to experiment without committing noise to the current branch.

## Key commands and examples

### 1) git stash push (preferred)

Save your local modifications to a new stash entry and revert working tree to HEAD.

Command:

```
git stash push -m "message describing work"
```

Example:

```
git stash push -m "WIP: navbar responsive fixes"
```

What happens:

- Staged and unstaged changes are saved in the stash.
- Working tree is reset to match HEAD so you can switch branches cleanly.

### 2) git stash list

Show stash entries in your repository.

Command:

```
git stash list
```

Example output:

```
stash@{0}: On main: WIP: navbar responsive fixes  
stash@{1}: On feature-x: WIP: try new auth flow
```

### 3) git stash show

Show summary of changes in a stash entry.

Command:

```
git stash show stash@{0}
```

For full diff:

```
git stash show -p stash@{0}
```

### 4) git stash apply

Re-apply the changes from a stash entry to your working directory but keep the stash in the stash list.

Command:

```
git stash apply stash@{0}
```

Typical use:

```
git stash apply # applies stash@{0}
```

Notes:

- If conflicts occur, resolve them like normal merge conflicts.
- The stash remains; delete it manually when you no longer need it.

## 5) git stash pop

Apply the stash and remove it from the stash list (apply + drop).

Command:

```
git stash pop stash@{0}
```

Example:

```
git stash pop
```

Notes:

- If the apply step conflicts, the stash may still be removed. Be cautious if you want to keep the stash on conflicts.

## 6) git stash drop

Remove a single stash entry.

Command:

```
git stash drop stash@{1}
```

Or drop the latest:

```
git stash drop
```

## 7) git stash clear

Remove all stash entries. Use with caution.

Command:

```
git stash clear
```

## 8) git stash branch

Create and check out a new branch starting from the commit where the stash was created, then apply the stash to it and remove the stash if successful.

Command:

```
git stash branch <new-branch-name> stash@{0}
```

Example:

```
git stash branch feature/navbar-fix stash@{0}
```

This is useful when you realize your local changes deserve their own branch.

## 9) Stashing untracked or ignored files

By default, `git stash` does not stash untracked files. Use options:

- u or --include-untracked : include untracked files
- a or --all : include ignored files as well

Example:

```
git stash push -u -m "WIP with new assets"
```

## 10) Creating a named stash (message)

Always add a short message to describe the stash so future-you knows what it contains:

```
git stash push -m "WIP: component X refactor"
```

Practical scenarios

Scenario A — switch branch to fix a hotbug:

1. `git stash push -m "WIP: messy layout"`
2. `git checkout hotfix/urgent`
3. fix bug, commit
4. `git checkout main`
5. `git stash pop` # re-apply your WIP

Scenario B — create a new branch from uncommitted work:

1. `git stash push -m "WIP: new auth"`
2. `git stash branch feature/auth-from-wip stash@{0}`  
(this creates branch and applies stash)

## Troubleshooting & tips

- If ``git stash apply`` shows conflicts, use ``git status`` and resolve conflicts, then ``git add`` and
- Use ``git stash show -p`` to inspect before applying.
- Remember: stash is local to the repository (and not pushed to remotes).
- Use clear, descriptive messages for stashes.

## Cheatsheet (commands)

<code>git stash push -m "message"</code>	# save changes with message
<code>git stash list</code>	# list stash entries
<code>git stash show [-p] stash@{n}</code>	# show summary or full diff
<code>git stash apply [stash@{n}]</code>	# apply without removing stash
<code>git stash pop [stash@{n}]</code>	# apply and remove stash
<code>git stash drop [stash@{n}]</code>	# remove a stash entry
<code>git stash clear</code>	# remove all stash entries
<code>git stash branch &lt;new-branch&gt; [n]</code>	# create branch from stash and apply
<code>git stash push -u</code>	# include untracked files
<code>git stash push -a</code>	# include ignored files

## Further reading

Official docs and ``git help stash`` are excellent next steps.