Hands-on guide for Git Rebase flow, resolving conflicts, and viewing history

Step-by-step **Hands-on guide for Git Rebase flow, resolving conflicts, and viewing history** (assuming Bitbucket Enterprise with personal token auth):

Step 1: Setup Local Repository

- Clone your repository from Bitbucket Enterprise:

```
git clone https://bitbucket.yourcompany.com/scm/project/repo.git
cd repo

```

Step 2: Create a new branch to work on

- Create and switch to a feature branch:

```
git checkout -b feature-branch

```

Step 3: Make changes and commit

- Edit files locally, then add and commit changes:

```
git add .
git commit -m "Feature work in progress"

```

Step 4: Fetch latest changes from main branch

- Fetch updates and switch to main:

```
git fetch origin
git checkout main
git pull origin main

```

Step 5: Rebase your feature branch on top of updated main

- Switch back to feature branch and rebase:

```
git checkout feature-branch
git rebase main

```

Step 6: Resolve any conflicts during rebase

- If conflicts occur, Git will pause and show conflicted files.
- Open conflicted files, manually edit to resolve conflicts.
- After resolving conflicts, stage the resolved files:

```
git add <file-name>

```

- Continue the rebase:

```
git rebase --continue

```

- Repeat until rebase finishes.

Step 7: View Git history

- To see commit history (linear after rebase):

```
git log --oneline --graph --decorate

```

- Or use a GUI tool like `gitk` or VSCode Git panel.

Step 8: Push rebased branch to Bitbucket Enterprise

- Because of rebase, you need to force push:

```
git push origin feature-branch --force-with-lease

```

Notes:

- Always use `--force-with-lease` instead of `--force` for safer force push.
- Make sure no one else is working on the same branch before force pushing.
- Use Bitbucket Pull Requests to review code after pushing your rebased branch.