# Debugging software with git bisect

```
git bisect start [--term-{new,bad}=<term> --term-{old,good}=<term>]
      [--no-checkout] [--first-parent] [<bad> [<good>...]] [--] [<paths>...]
git bisect (bad|new|<term-new>) [<rev>]
git bisect (good|old|<term-old>) [<rev>...]
git bisect terms [--term-good | --term-bad]
git bisect skip [(<rev>|<range>)...]
git bisect reset [<commit>]
git bisect (visualize|view)
git bisect replay <logfile>
git bisect log
git bisect run <cmd>...
git bisect help
```

git bisect uses a binary search algorithm to find which commit in your project's history introduced a bug.

- `$> git bisect start`

- `$> git bisect start`
- `$> git bisect bad # Current version is bad`

- `$> git bisect start`
- `$> git bisect bad # Current version is bad`
- `$> git bisect good # Current version is good`

Let's take it for a spin!

# What we're going to do:

# What we're going to do:

- 👀 Check the version history

# What we're going to do:

- 👀 Check the version history
- 🧙‍♀️ start a **git bisect** section & find the bug

# What we're going to do:

- 👀 Check the version history
- 🧙‍♀️ start a **git bisect** section & find the bug
- 🎉 Fix the bug

# In Summary

# In Summary

- **git bisect** allows you to proceed by elimination to pinpoint the version that introduced an issue

# In Summary

- **git bisect** allows you to proceed by elimination to pinpoint the version that introduced an issue
- It works very well when your codebase has **small, understandable** commit messages

# Thanks!