title: Quick Tips

Here is a collection of tips for working with Goose:

**Goose can and will edit files**

Use a git strategy to avoid losing anything - such as staging your personal edits and leaving Goose edits unstaged until reviewed. Consider using individual commits which can be reverted. See more on [File Management](/docs/guides/file-management).

**Goose can and will run commands**

Goose will check commands for safety, however, you can also ask Goose to check with you before running commands if you are concerned.

**You can interrupt Goose with CTRL+C**

Use this command to stop Goose in the event that you need to correct it or give it more info.

**Goose works best when solving concrete problems**

Experiment with how far you need to break a problem down to get Goose to solve it. Be specific! For example, it will likely fail to "create a banking app", but probably does a good job if prompted with "create a Fastapi app with an endpoint for deposit and withdrawal and with account balances stored in mysql keyed by id".

**Goose can troubleshoot**

If something goes wrong, Goose can help troubleshoot issues by examining logs, analyzing error messages, and suggesting possible resolutions.

**Leverage Goose to learn**

Use Goose to learn new technologies or frameworks by asking it to explain things like code snippets, concepts, or best practices relevant to your project.

**Goose needs context**

If Goose doesn't have enough context to start with, it might go in the wrong direction. Tell it to read files that you are referring to or search for objects in code. Even better, ask it to summarize them for you, which will help it set up its own next steps. You can create a [goosehints](/docs/guides/using-goosehints) file to help with this.

**Use easy search terms**

Refer to any objects in files with something that is easy for Goose to search for, such as "the MyExample class".

**Teach Goose how you test**

Goose **loves** to know how to run tests to get a feedback loop going, just like you do. If you tell it how you test things locally and quickly, it can make use of that when working on your project.

**Goose can do scripting tasks**

You can use Goose for tasks that would require scripting. It can even look at your screen and correct designs, or help you fix bugs. Try asking it to help you in a way you would ask a person.

**Goose will make mistakes**

Sometimes Goose will go in the wrong direction. Feel free to correct it, or start over again.

**Goose can run tasks continuously if asked**

You can tell Goose to run things for you continuously and it will iterate, try, and retry.

**Goose can run anywhere**

It doesn't have to be in a repo. Just ask Goose!

**Keep Goose updated**

Regularly update Goose to benefit from the latest features, bug fixes, and performance improvements. For the CLI, the best way to keep it updated is by re-running the [Goose installation script](https://block.github.io/goose/docs/quickstart/#installation). For Goose Desktop, check the [GitHub Releases page](https://github.com/block/goose/releases/stable) regularly for updates.

**title: Using Goosehints**

sidebar\_position: 3

**Providing Hints to Goose**

.goosehints is a text file used to provide additional context about your project and improve the communication with Goose. The use of goosehints ensures that Goose understands your requirements better and can execute tasks more effectively.

:::info Developer extension required  
To make use of the hints file, you need to have the developer extension [enabled](https://www.notion.so/docs/getting-started/using-extensions).

:::

This guide will walk you through creating and using your .goosehints file to streamline your workflow with custom instructions and context.

**Creating your hints file**

Create a file named .goosehints and save the file in ~/.config/goose/.goosehints. If saved here, Goose will use this file for every session with you.

:::tip  
You can also save .goosehints local to any directory. In this case, Goose will utilize the hints when working in that directory.  
:::

The .goosehints file can include any instructions or contextual details relevant to your projects.

A good time to consider adding a .goosehints file is when you find yourself repeating prompts, or providing the same kind of instructions multiple times. It's also a great way to provide a lot of context which might be better suited in a file.

**Setting up hints**

The .goosehints file supports natural language and also follows [jinja templating rules](https://jinja.palletsprojects.com/en/stable/), so you can leverage templating to insert file contents or variables.

Here are some ways people have used hints to provide additional context to Goose:

* **Decision-Making**: Specify if Goose should autonomously make changes or confirm actions with you first.
* **Validation Routines**: Provide test cases or validation methods that Goose should perform to ensure changes meet project specifications.
* **Feedback Loop**: Include steps that allow Goose to receive feedback and iteratively improve its suggestions.
* **Point to more detailed documentation**: Indicate important files like README.md, CONTRIBUTING.md, or others that Goose should consult for detailed explanations.

Like prompts, this is not an extensive list to shape your .goosehints file. You can include as much context as you need.

Example .goosehints file:

This is a simple example JavaScript web application that uses the Express.js framework. View [Express documentation](<https://expressjs.com/>) for extended guidance.  
  
Go through the README.md for information on how to build and test it as needed.  
  
Make sure to confirm all changes with me before applying.  
  
Use the following custom values when needed:  
{%include custom-config.js%}  
  
Run tests with `npm run test` ideally after each change.

**Best practices**

* **Keep file updated**: Regularly update the .goosehints file to reflect any changes in project protocols or priorities.
* **Be concise**: Make sure the content is straightforward and to the point, ensuring Goose can quickly parse and act on the information.

**title: Using Goosehints**

sidebar\_position: 3

**Providing Hints to Goose**

.goosehints is a text file used to provide additional context about your project and improve the communication with Goose. The use of goosehints ensures that Goose understands your requirements better and can execute tasks more effectively.

:::info Developer extension required  
To make use of the hints file, you need to have the developer extension [enabled](https://www.notion.so/docs/getting-started/using-extensions).

:::

This guide will walk you through creating and using your .goosehints file to streamline your workflow with custom instructions and context.

**Creating your hints file**

Create a file named .goosehints and save the file in ~/.config/goose/.goosehints. If saved here, Goose will use this file for every session with you.

:::tip  
You can also save .goosehints local to any directory. In this case, Goose will utilize the hints when working in that directory.  
:::

The .goosehints file can include any instructions or contextual details relevant to your projects.

A good time to consider adding a .goosehints file is when you find yourself repeating prompts, or providing the same kind of instructions multiple times. It's also a great way to provide a lot of context which might be better suited in a file.

**Setting up hints**

The .goosehints file supports natural language and also follows [jinja templating rules](https://jinja.palletsprojects.com/en/stable/), so you can leverage templating to insert file contents or variables.

Here are some ways people have used hints to provide additional context to Goose:

* **Decision-Making**: Specify if Goose should autonomously make changes or confirm actions with you first.
* **Validation Routines**: Provide test cases or validation methods that Goose should perform to ensure changes meet project specifications.
* **Feedback Loop**: Include steps that allow Goose to receive feedback and iteratively improve its suggestions.
* **Point to more detailed documentation**: Indicate important files like README.md, CONTRIBUTING.md, or others that Goose should consult for detailed explanations.

Like prompts, this is not an extensive list to shape your .goosehints file. You can include as much context as you need.

Example .goosehints file:

This is a simple example JavaScript web application that uses the Express.js framework. View [Express documentation](<https://expressjs.com/>) for extended guidance.  
  
Go through the README.md for information on how to build and test it as needed.  
  
Make sure to confirm all changes with me before applying.  
  
Use the following custom values when needed:  
{%include custom-config.js%}  
  
Run tests with `npm run test` ideally after each change.

**Best practices**

* **Keep file updated**: Regularly update the .goosehints file to reflect any changes in project protocols or priorities.
* **Be concise**: Make sure the content is straightforward and to the point, ensuring Goose can quickly parse and act on the information.