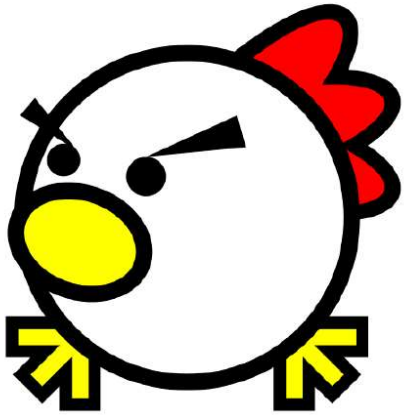
Creating the Vim Version of VSCode Dev Container Extension: Why and How

mikoto2000

About

Me:

- Name: mikoto2000
- GitHub: https://github.com/mikoto2000
- X(Twitter): https://x.com/mikoto2000



I creating some Tauri applications with devcontainer.vim.

Session Topic Repository:
https://github.com/mikoto2000/devcontainer.vim

Contents:

- What did I make?
- Why did I make?
- What problem did we solve, and how did we do it?

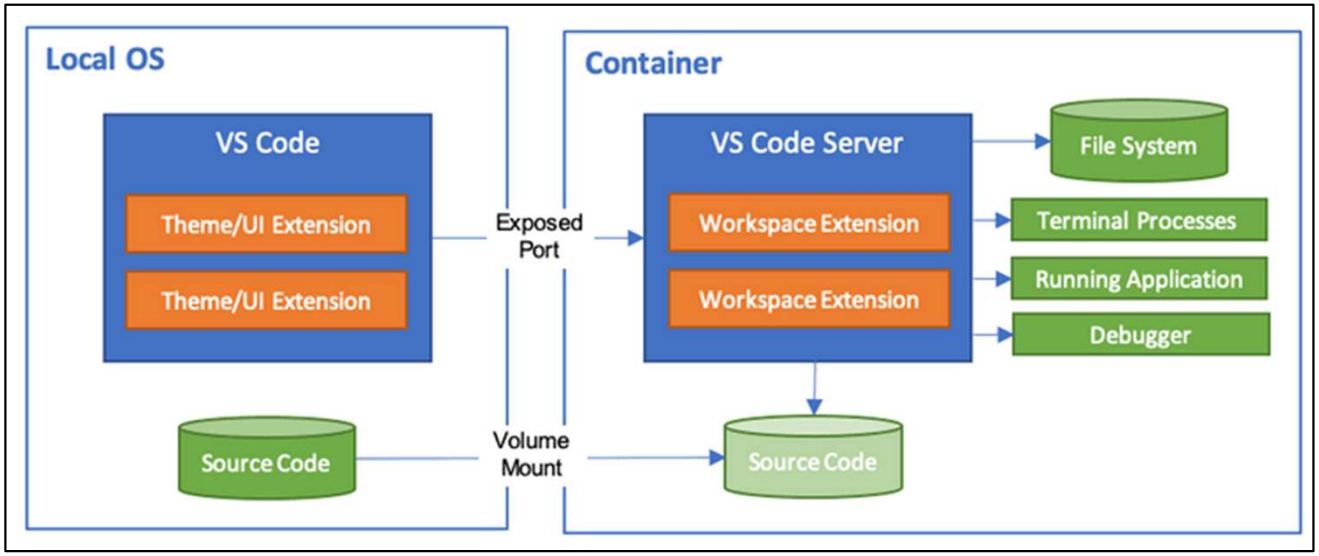
What did I make?

Contents:

- Prerequisites: What is Visual Studio Code Dev Container extension
- Prerequisites: devcontainers/cli
- Created by: devcontainer.vim(Vim Version of VSCode Dev Container Extension)

What is Visual Studio Code Dev Container extension(1/2)

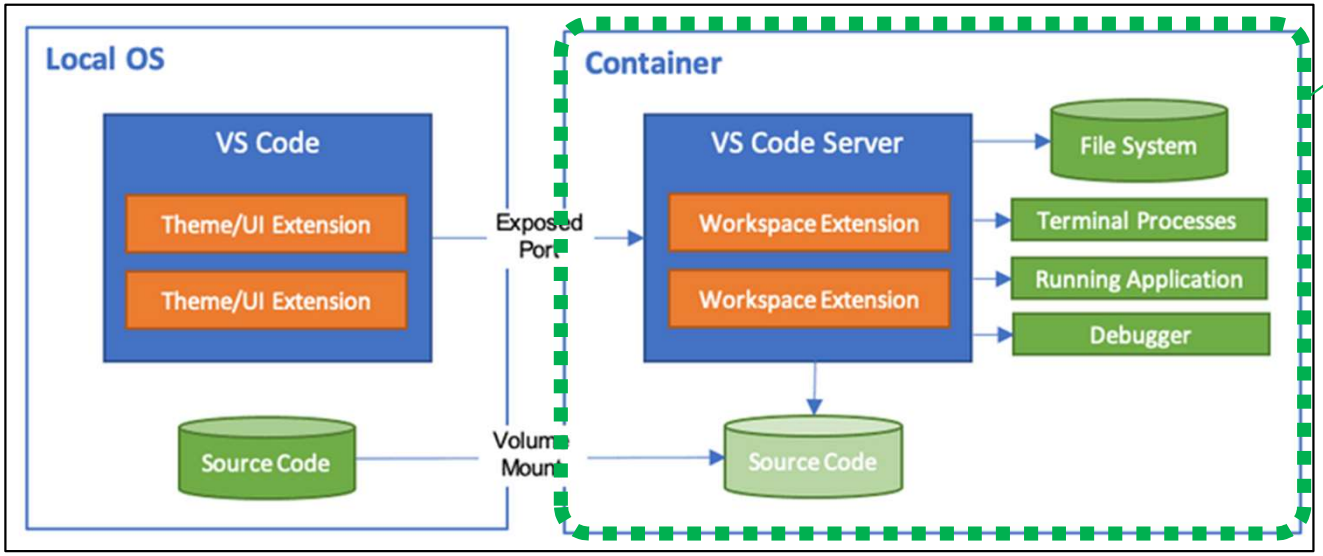
- Start the container as described in `.devcontainer`, deploy Visual Studio Code to the started development container, and then run it.
- Only the UI of the development container is sent to the host, and operations are performed on the host.
- The operations performed on the host are sent to VS Code running in the container, and the actual operations on the development target are performed on the container.



† Developing inside a Container using Visual Studio Code Remote Development from https://code.visualstudio.com/docs/devcontainers/containers

What is Visual Studio Code Dev Container extension(2/2)

- Start the container as described in `.devcontainer`, deploy Visual Studio Code to the started development container, and then run it.
- Only the UI of the development container is sent to the host, and operations are performed on the host.
- The operations performed on the host are sent to VS Code running in the container, and the actual operations on the development target are performed on the container.



Create from `.devcontainer` files

```
PROJECT_ROOT/
    +- .devcontainer/
    |   +- devcontainer.json
    |
    +- src/
    +- (snip, other project files)
```

■ devcontainer.json

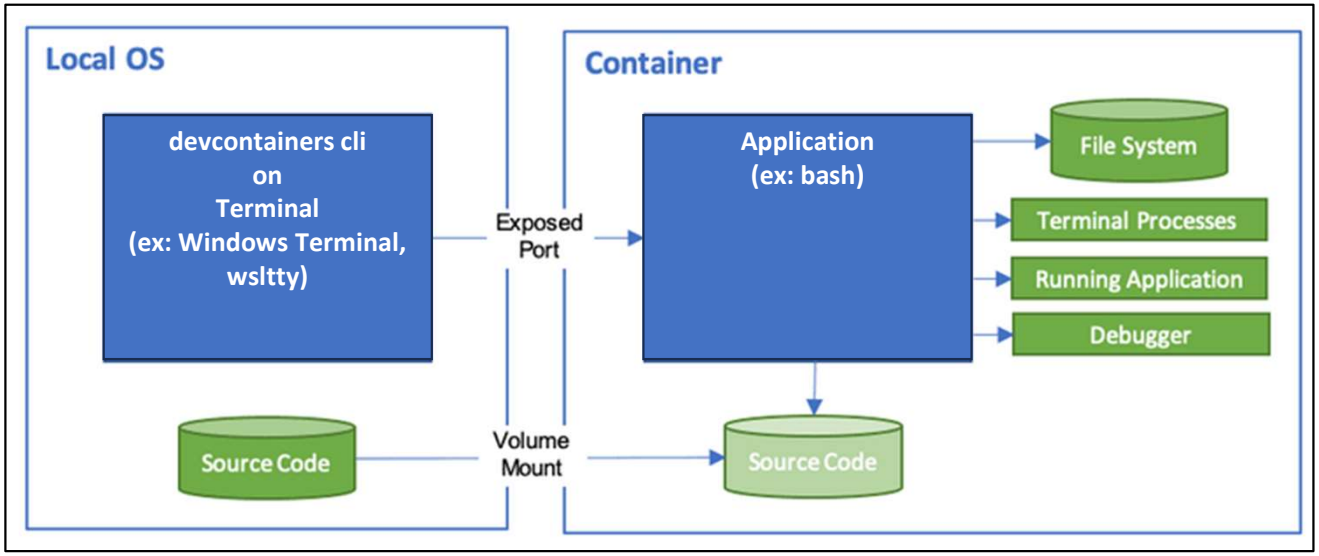
```
{
  name: "ruby",
  image: "ruby:3.3"
}
```

† Developing inside a Container using Visual Studio Code Remote Development from https://code.visualstudio.com/docs/devcontainers/containers

devcontainers/cli

Repository: https://github.com/devcontainers/cli

- CLI version of Visual Studio Code Dev Container extension.
- Only starts the container according to the configuration of `.devcontainer` and does not deploy VSCode.
- Basically, you enter the container using `devcontainer exec <container_name> bash` and work on the container.(Something like “VSCode has been removed from the VSCode Dev Container extension.)

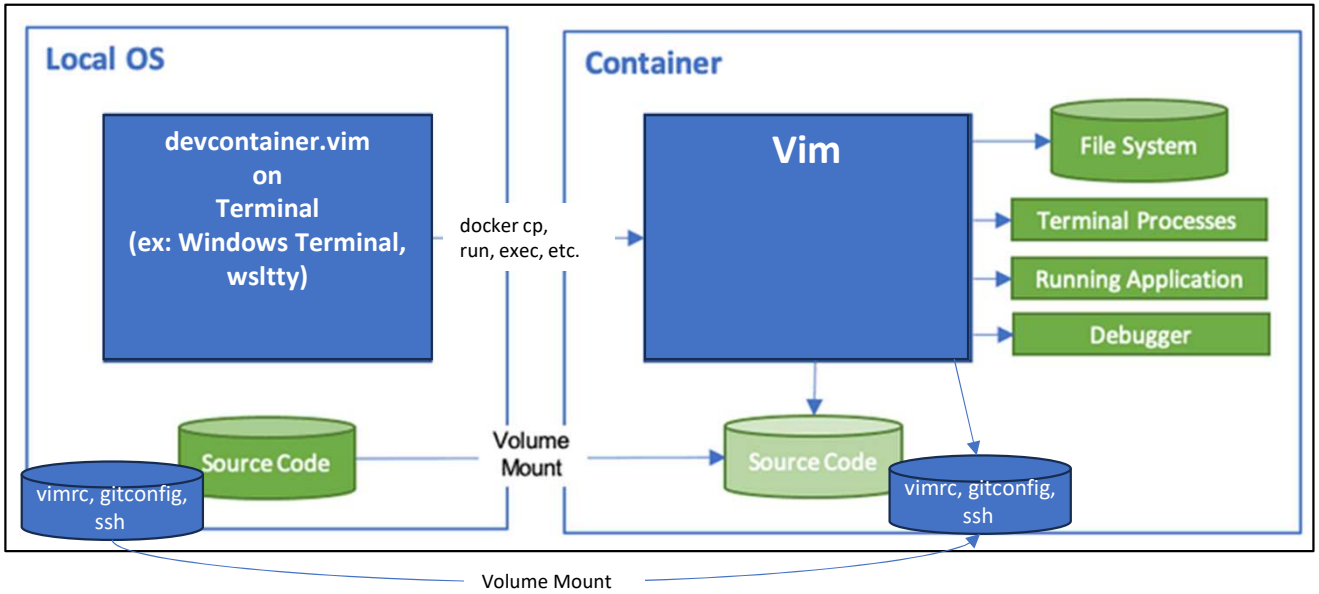


† Developing inside a Container using devcontainers/cli
Cited while modifying
https://code.visualstudio.com/docs/devcontainers/containers

What devcontainer.vim

Repository: https://github.com/mikoto2000/devcontainer.vim

- Run Vim instead of a shell in devcontainres/cli
- I say it's "Vim version of VSCode Dev Container extension".
- Send Vim to a container and provide a mechanism for all development to take place tucked inside the container
- Use bind mount for vimrc, gitconfig, ssh, etc. that you want to share with the host.



The process outline is as follows.

1. Start the container
2. Transfer Vim with `docker cp`
3. Launch Vim with `devcontainer exec`

I created a Go language program to automate these processes.

† Developing inside a Container using devcontainer.vim
Cited while modifying
https://code.visualstudio.com/docs/devcontainers/containers

Why did I make?

Contents:

- I wanted everything to be complete in the container
- The configuration that most people would first consider

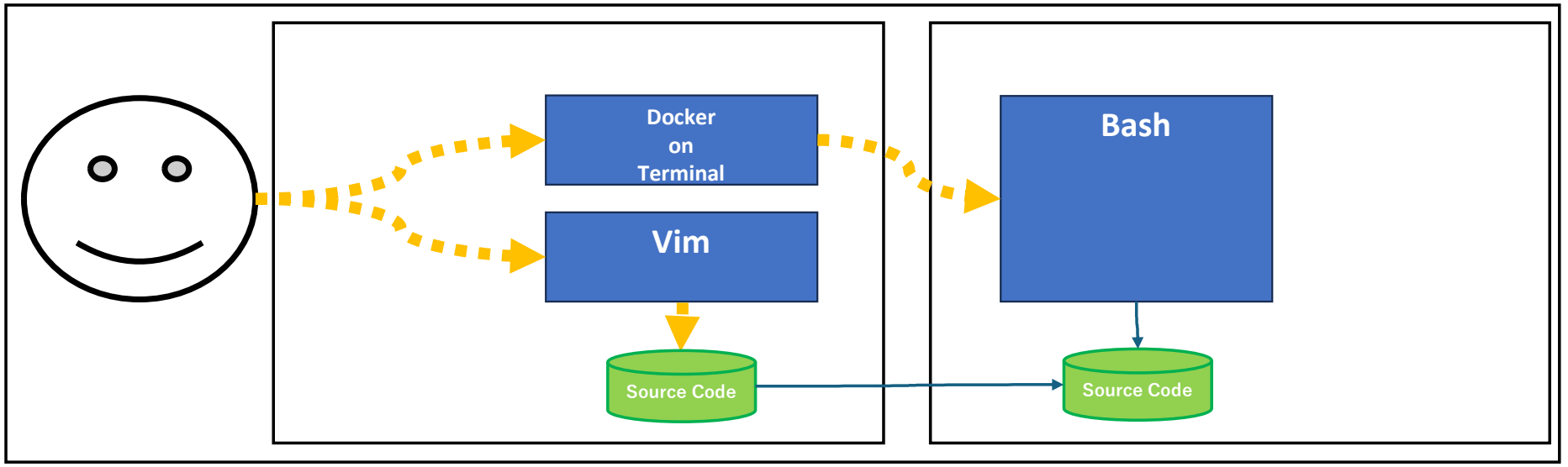
I wanted everything to be complete in the container.

- Container's pros
  - Easy environment distribution (for people using VSCode)
  - No need to worry about the host environment
- Container's cons
  - If it is not completed within the container, sufficient coding support cannot be obtained.

devcontainer.vim was created to address the shortcomings of this container.

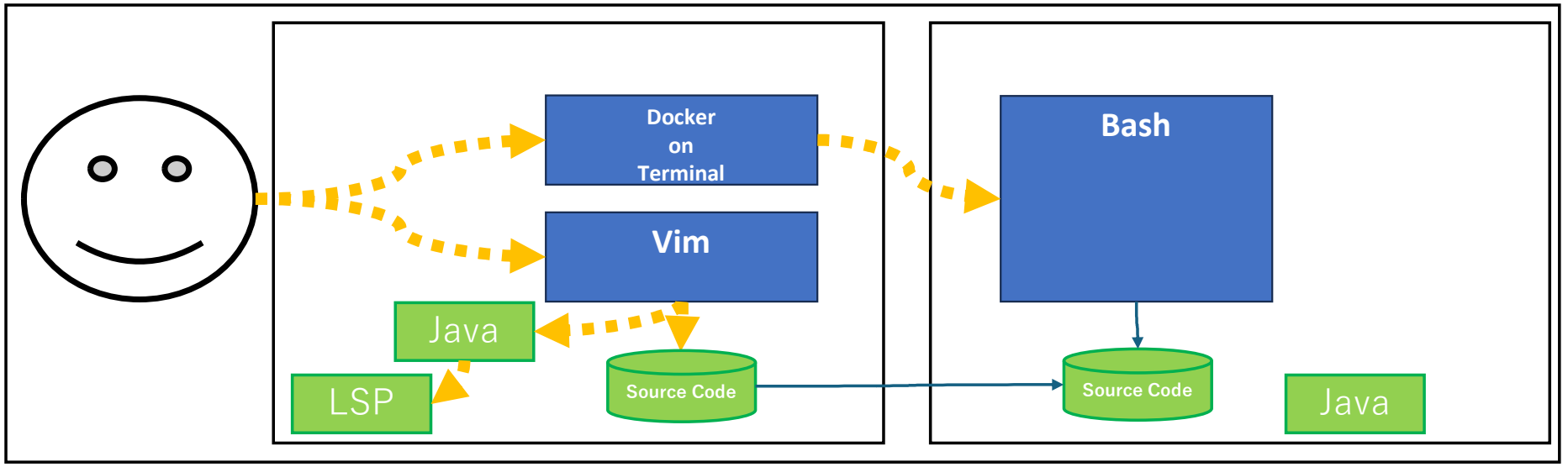
The configuration that most people would first consider(1/2)

- Coding, compiling, executing, and testing are all possible without problems.
- Coding assistance such as LSP is unavailable



The configuration that most people would first consider(2/2)

- When you try to use it, you need to build an environment on the Local OS, and you might wonder, "Why do I have to enter the host even though the environment is in the container?"
- To overcome this weakness and truly containerize, this tool was created.



Solved problems and methods

Contents:

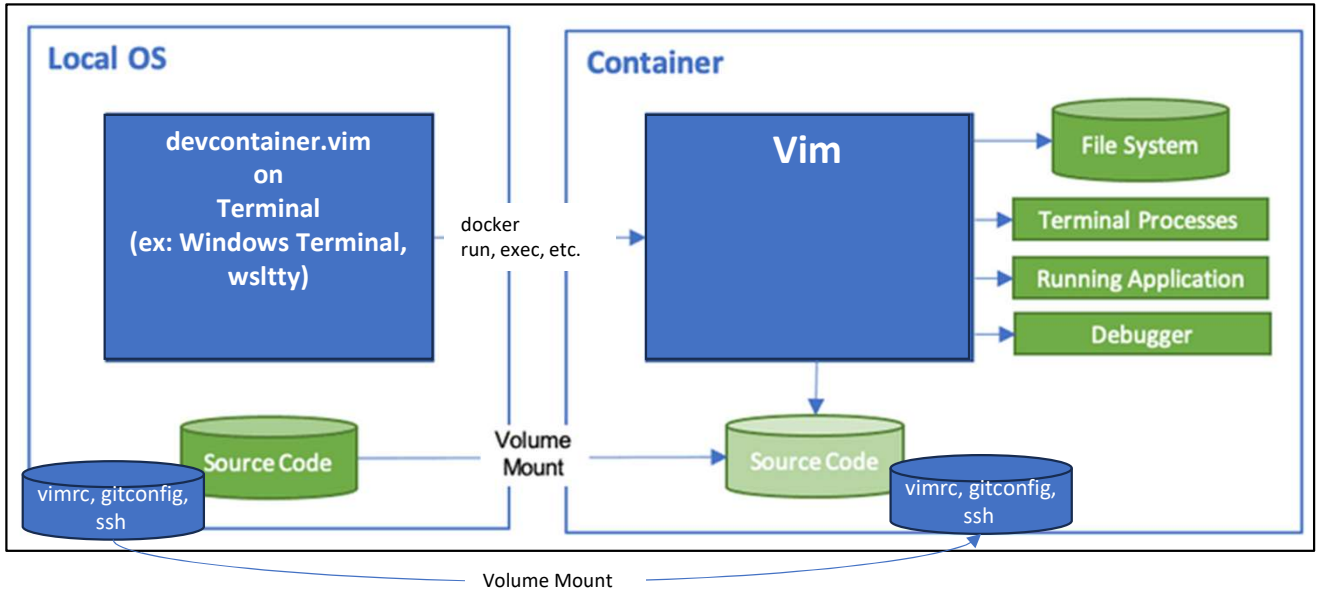
- Reprint of the composition of devcontainer.vim

- Weakness (1) of this method: Unnecessary definitions for people using VSCode
- Overcoming weakness (1): Consider people who use VSCode

- Weakness (2) of this method: Practical-level yank is not possible
- Overcoming Weakness (2): Send the yanked text via TCP and coordinate with the host

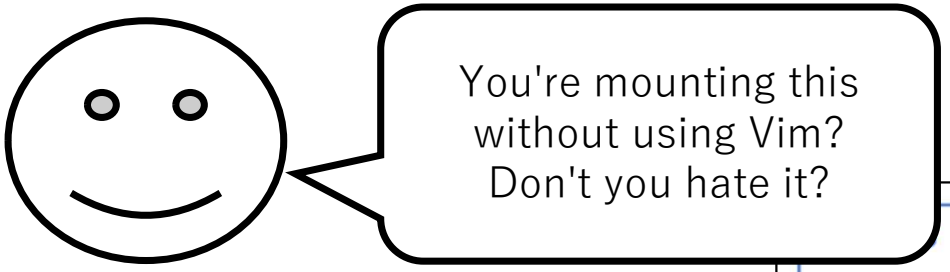
Reprint of the composition of devcontainer.vim

- Actually, this is a fairly simplified diagram, so I will elaborate on the issues that arose and how they were resolved.



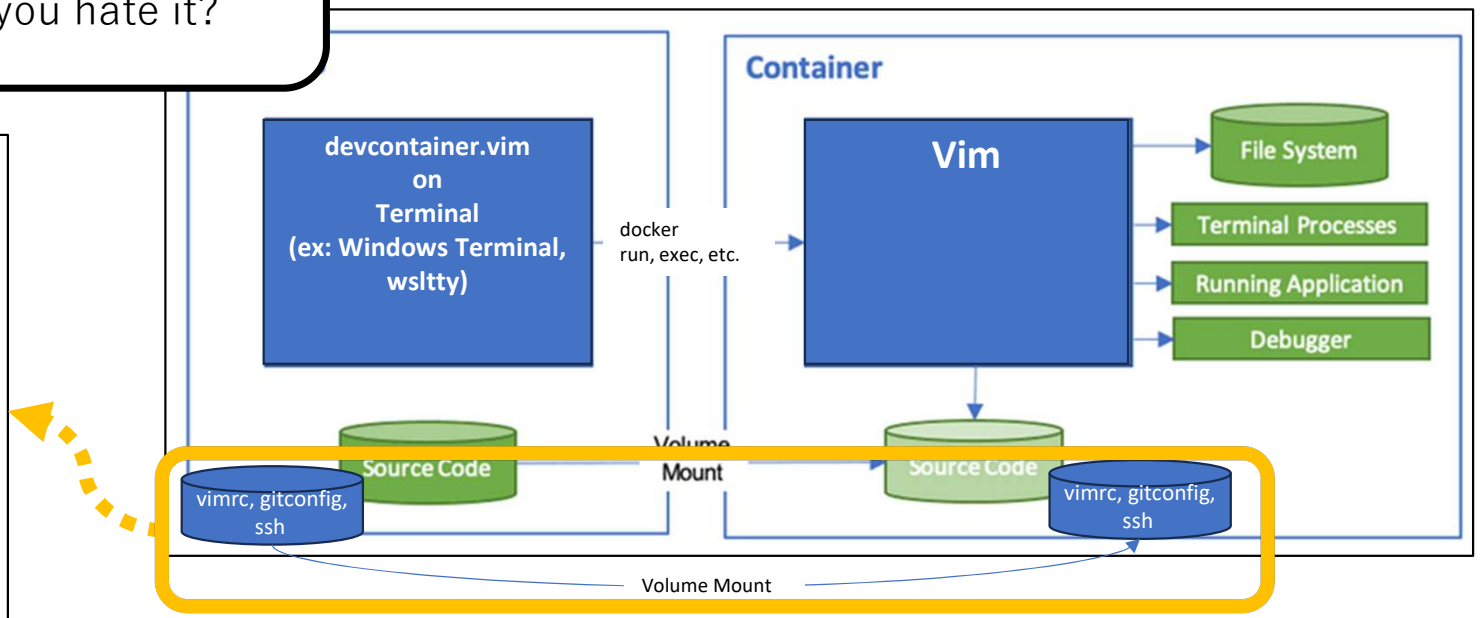
Weakness (1) of this method: Unnecessary definitions for people using VSCode

- Mount definitions for Vim are useless for existing Dev container users



■ .devcontainer/devcontainer.json

```
...(snip)
  {
      "type": "bind",
      "source": "${localEnv:HOME}/.vim",
      "target": "/home/vscode/.vim"
    },
    {
      "type": "bind",
      "source": "${localEnv:HOME}/.gitconfig",
      "target": "/home/vscode/.gitconfig"
    },
    {
      "type": "bind",
      "source": "${localEnv:HOME}/.ssh",
      "target": "/home/vscode/.ssh"
    },
...(snip)
```



Consider people who use VSCode

~/.cache/devcontainer.vim

■ Dev container 起動時に使われる json

```
{
  "name": "Go",
  "image": "go “,
  "mounts" : {
    {
      "type": "bind",
      "source": "${localEnv:HOME}/.vim",
      "target": "/home/vscode/.vim"
    },
    ...(snip)
  }
}
```

Define settings for VS Code (devcontainer.json), devcontainer.vim (devcontainer.vim.json) separately, and merge them just before starting the Dev container.

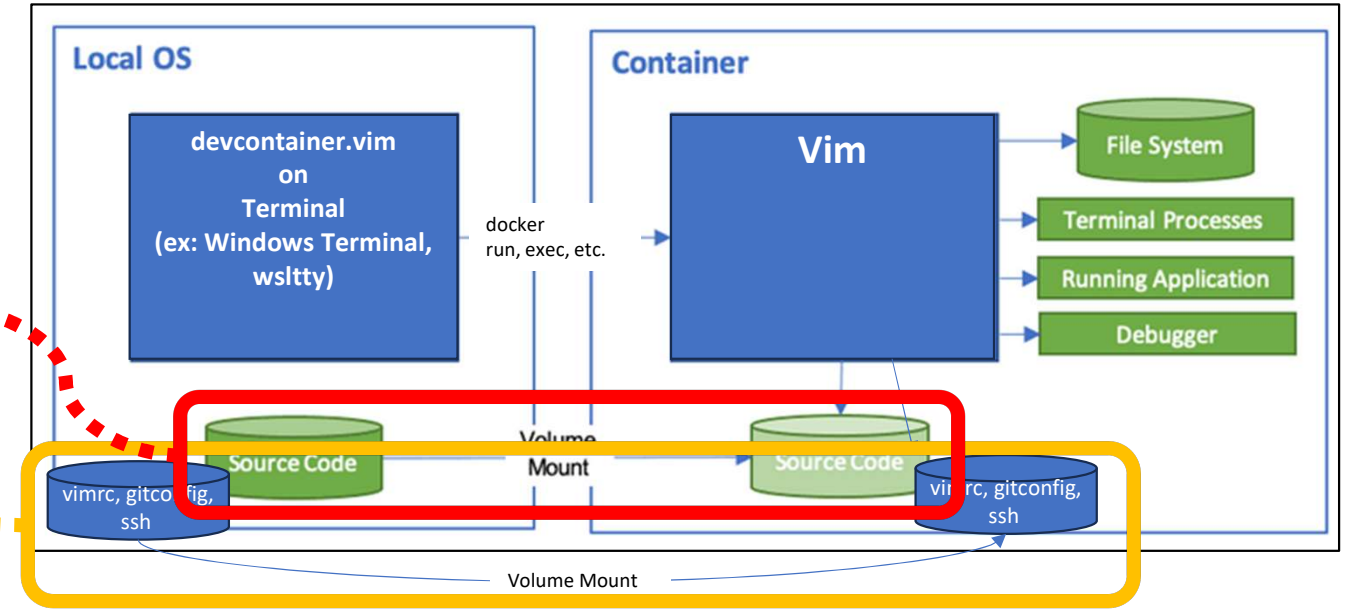
$PROJECT_ROOT/.devcontainer

■ devcontainer.vim.json

```
{
  {
    "type": "bind",
    "source": "${localEnv:HOME}/.vim",
    "target": "/home/vscode/.vim"
  },
  ...(snip)
  {
    "type": "bind",
    "source": "${localEnv:HOME}/.ssh",
    "target": "/home/vscode/.ssh"
  }
}
```

■ devcontainer.json

```
{
  "name": "Go",
  "image": "go"
}
```

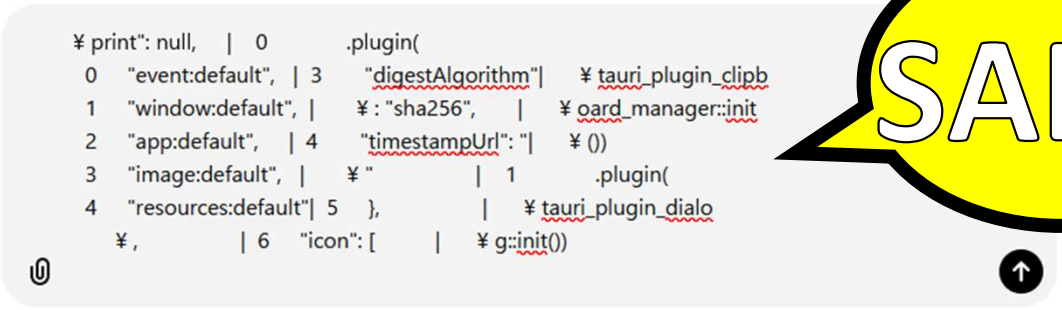


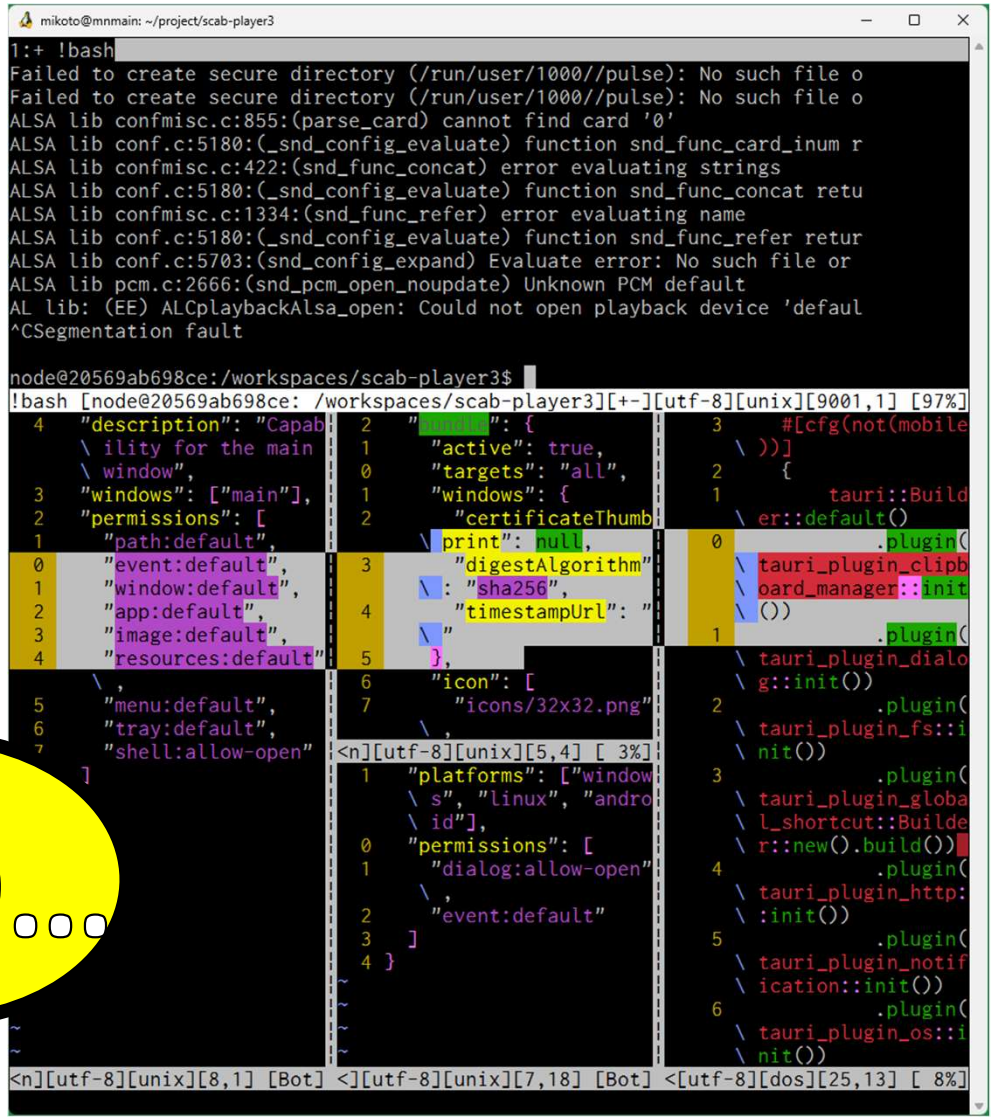
Weakness (2) Practical-level yank is not possible

- Cannot receive yanked data in Vim
- Copying terminal functions often results in disappointing outcomes...

vsplit is pierced and line numbers are copied...

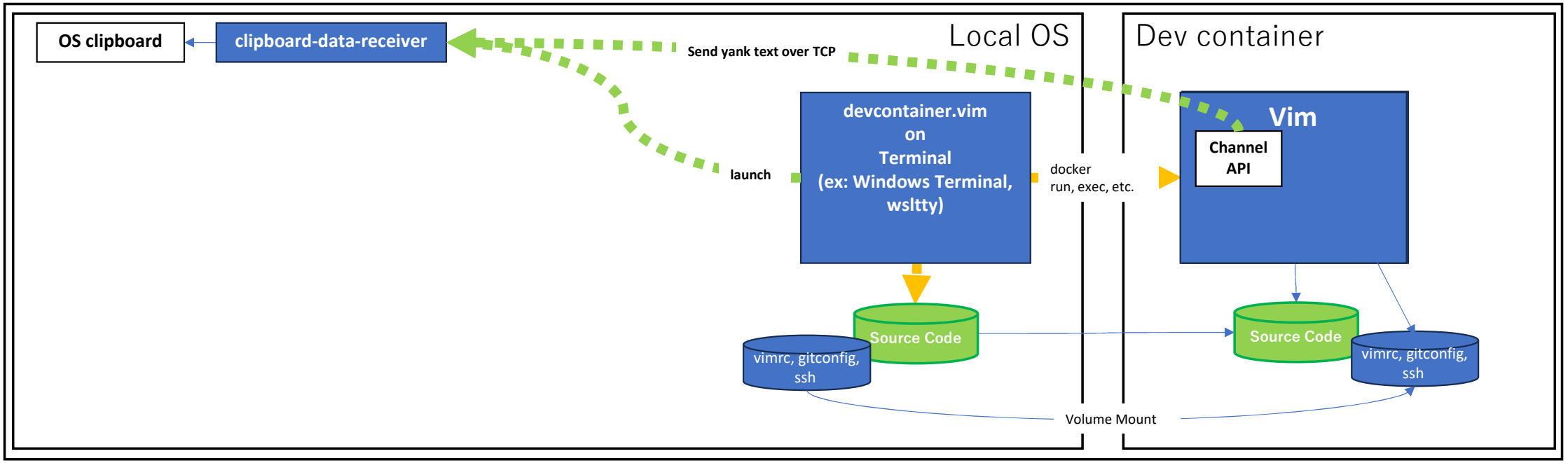
That would make it difficult to pass code snippets, documents, or error messages to Google or ChatGPT.





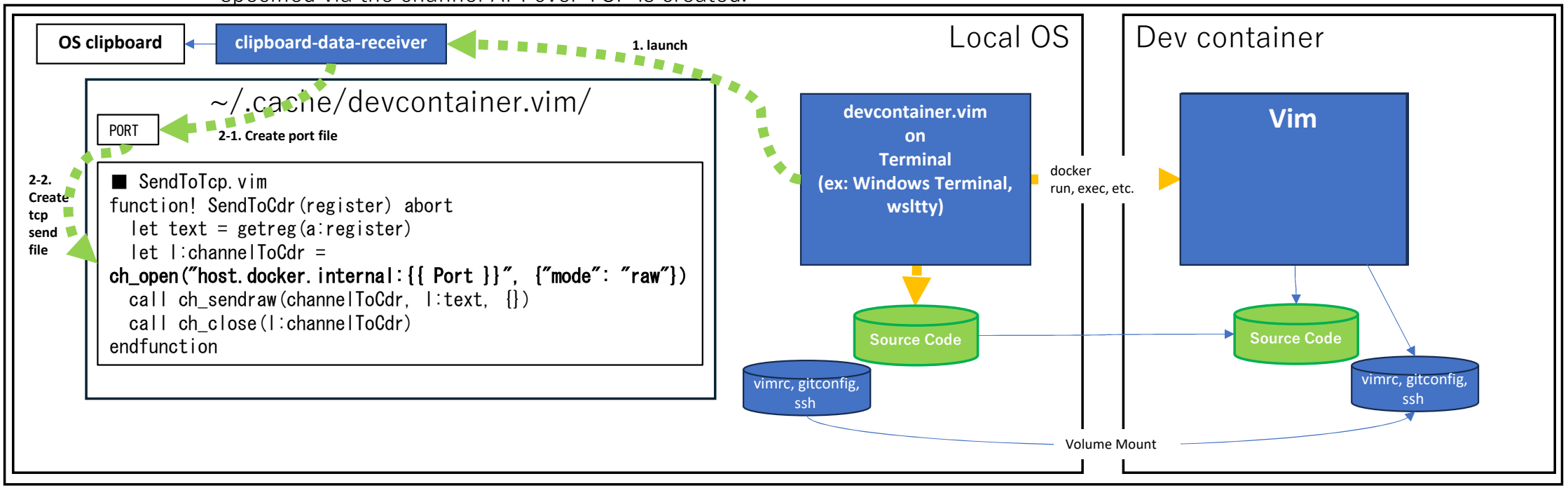
Send the yanked text via TCP and coordinate with the host(1/4)

- I created an app called `clipboard-data-receiver` so that the host can receive the string yanked from the container.
  - It runs on the host OS and listens on an arbitrary TCP port.
  - Reflect the string sent to the listen port on the host's clipboard
  - Vim to `clipboard-data-receiver` is achieved by TCP communication via Vim's channel.



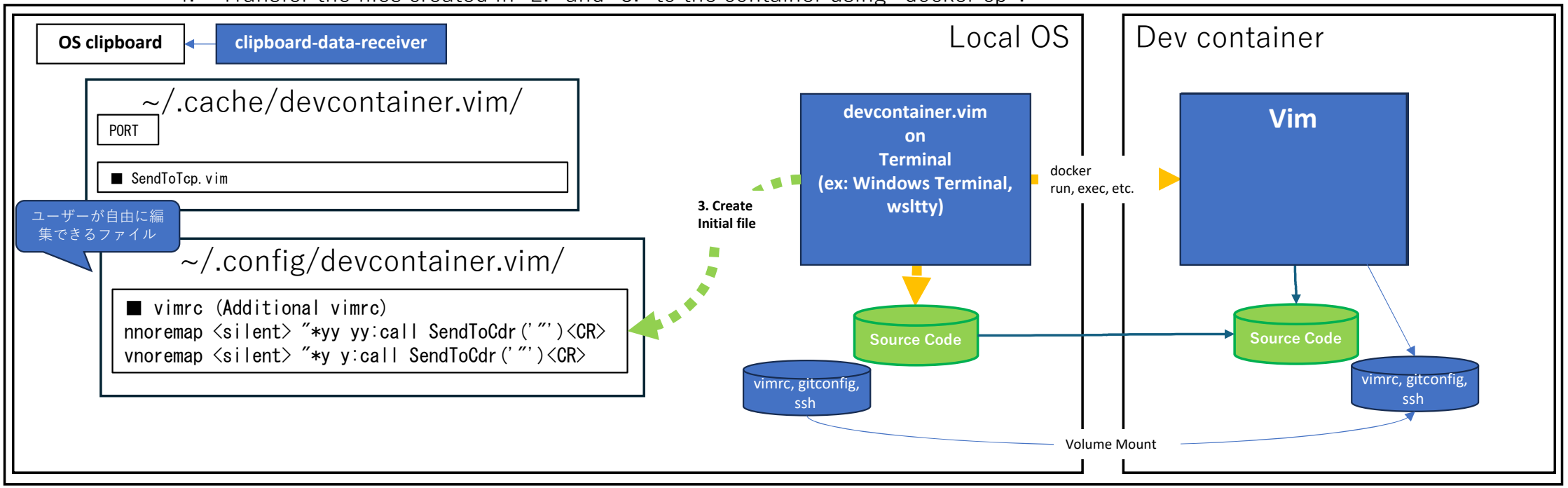
Send the yanked text via TCP and coordinate with the host(2/4)

- The process of receiving text data from Vim and reflecting it to the OS clipboard after starting `clipboard-data-receiver`.
  1. `devcontainer.vim` start `clipboard-data-receiver`.
  2. `clipboard-data-receiver` outputs a PORT file, so based on that, a function to send the text of the registers specified via the channel API over TCP is created.



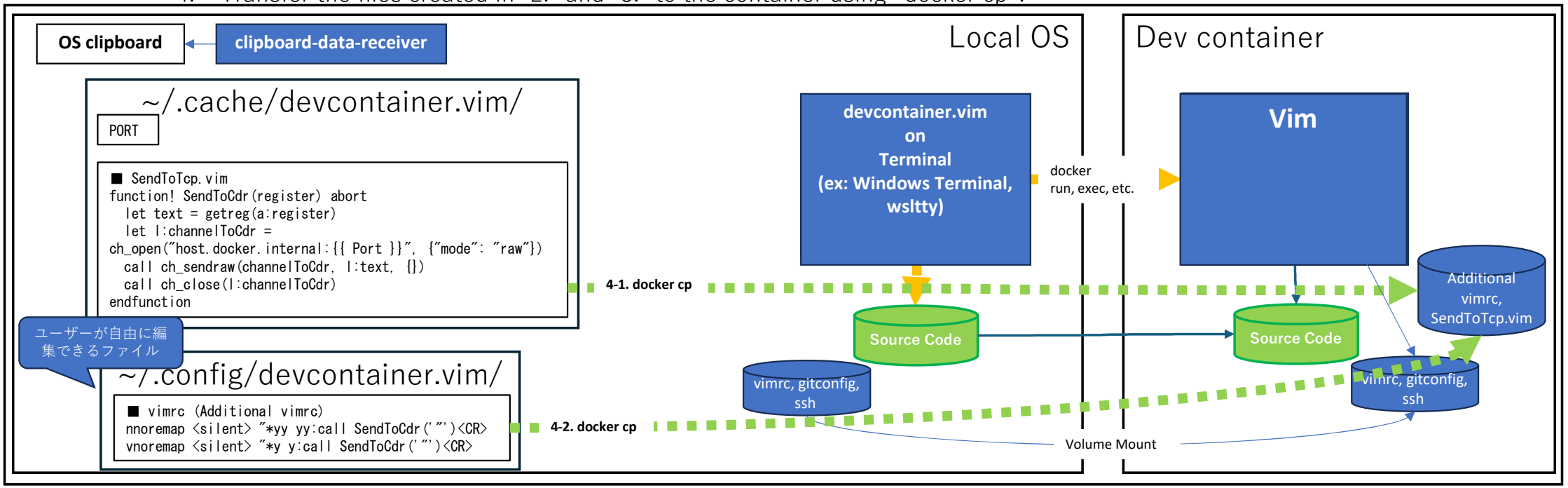
Send the yanked text via TCP and coordinate with the host(3/4)

- The process of receiving text data from Vim and reflecting it to the OS clipboard after starting `clipboard-data-receiver`.
  3. Create a mapping file that is only active when `devcontainer.vim` is started (by default, a mapping that launches the function created in step 2 is generated).
  4. Transfer the files created in "2." and "3." to the container using `docker cp`.



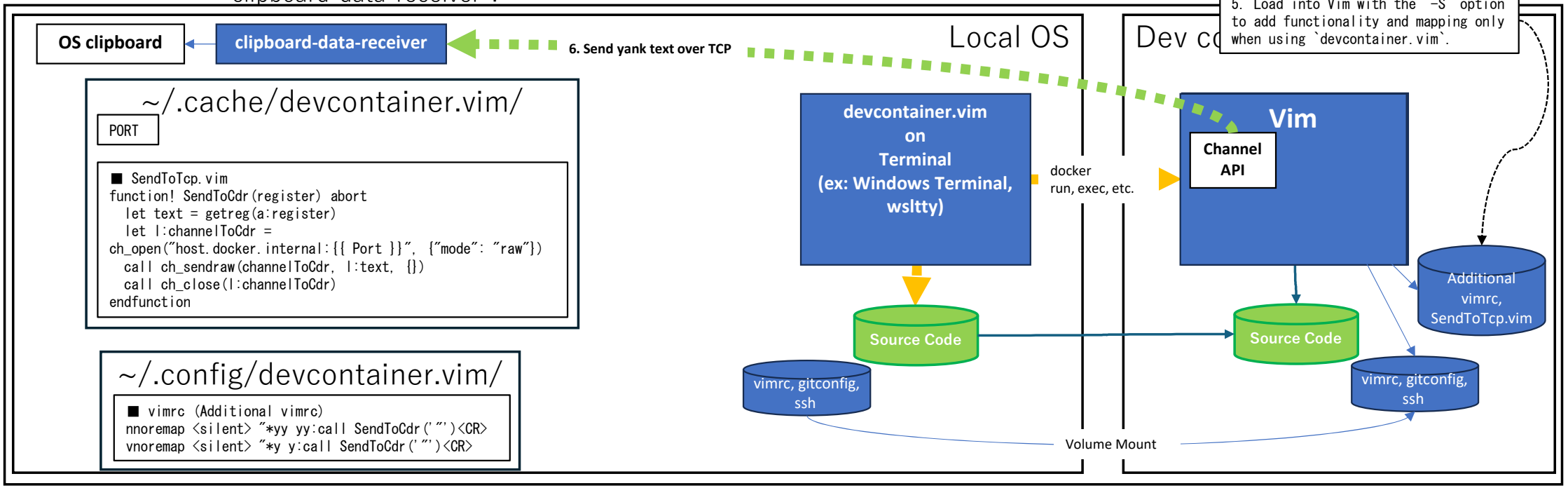
Send the yanked text via TCP and coordinate with the host(3/4)

- The process of receiving text data from Vim and reflecting it to the OS clipboard after starting `clipboard-data-receiver`.
  3. Create a mapping file that is only active when `devcontainer.vim` is started (by default, a mapping that launches the function created in step 2 is generated).
  4. Transfer the files created in "2." and "3." to the container using `docker cp`.



Send the yanked text via TCP and coordinate with the host(4/4)

- The process of receiving text data from Vim and reflecting it to the OS clipboard after starting `clipboard-data-receiver`.
  5. Read the file transferred with the `-S` option when Vim starts.
  6. (By default) When yanked, calls the SendToCdr function with the `"` register, which sends the yanked text to the `clipboard-data-receiver`.



Summary

- I created a tool that integrates Vim with Dev containers, allowing me to develop in a Dev container.
  - The practical application was achieved by solving two problems.

- It's difficult but possible to integrate Vim with some environments
  - Console apps work in most environments.
  - Embed dedicated processing for the integration target with `-S`.
  - The Channel API enables integration between the target and the host.

- The future of devcontainer.vim
  - Try to use statically linked binaries instead of AppImage.