# AI-assisted coding tools workshop: With Cline

## What is Cline?

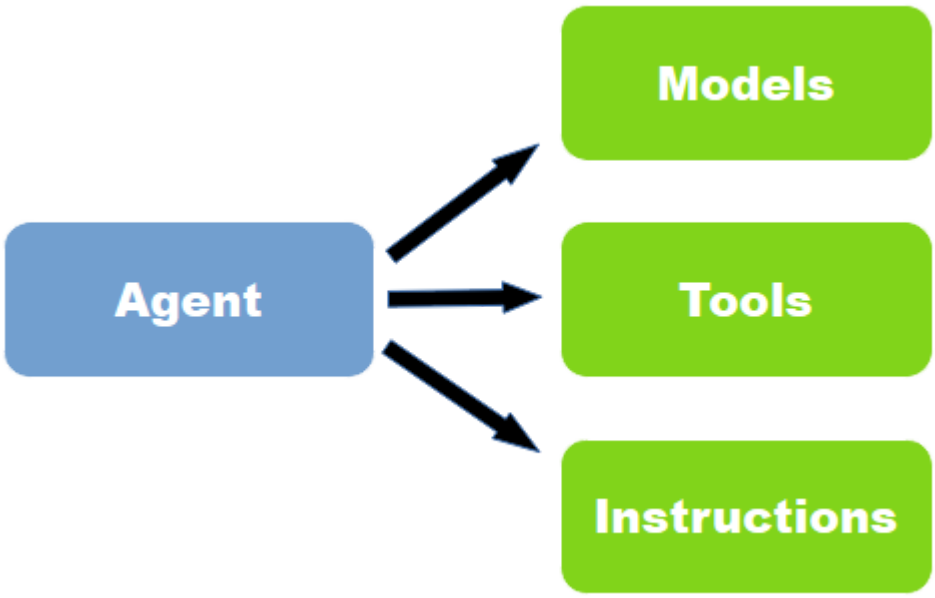
Cline is an autonomous, open-source coding agent

- Cline represents a new generation of AI-powered development tools that can understand complex coding tasks, plan solutions, and implement them with minimal human intervention. It integrates seamlessly with VS Code to provide a comprehensive coding assistant experience.

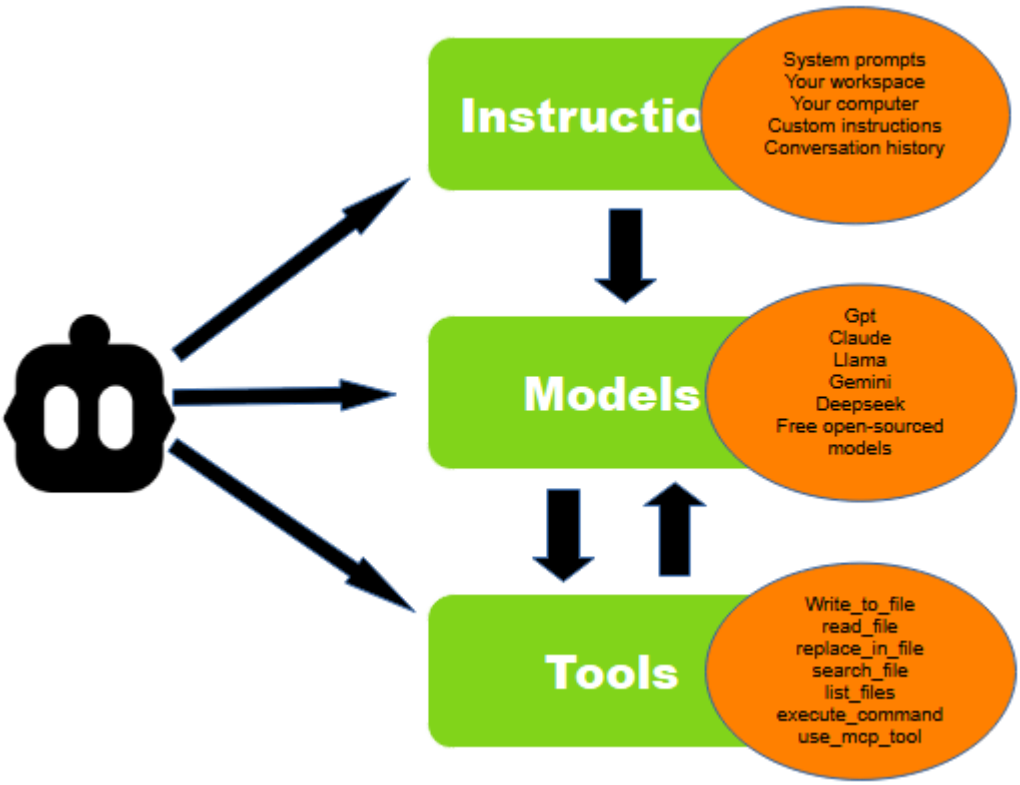
## What is an Agent?

- **OpenAI** described agents as systems that work autonomously to accomplish tasks on your behalf
- **Anthropic** described agents as systems where LLMs dynamically direct their own processes and tool usage, maintaining control over how they accomplish tasks

Agents are primarily composed of three main building components:



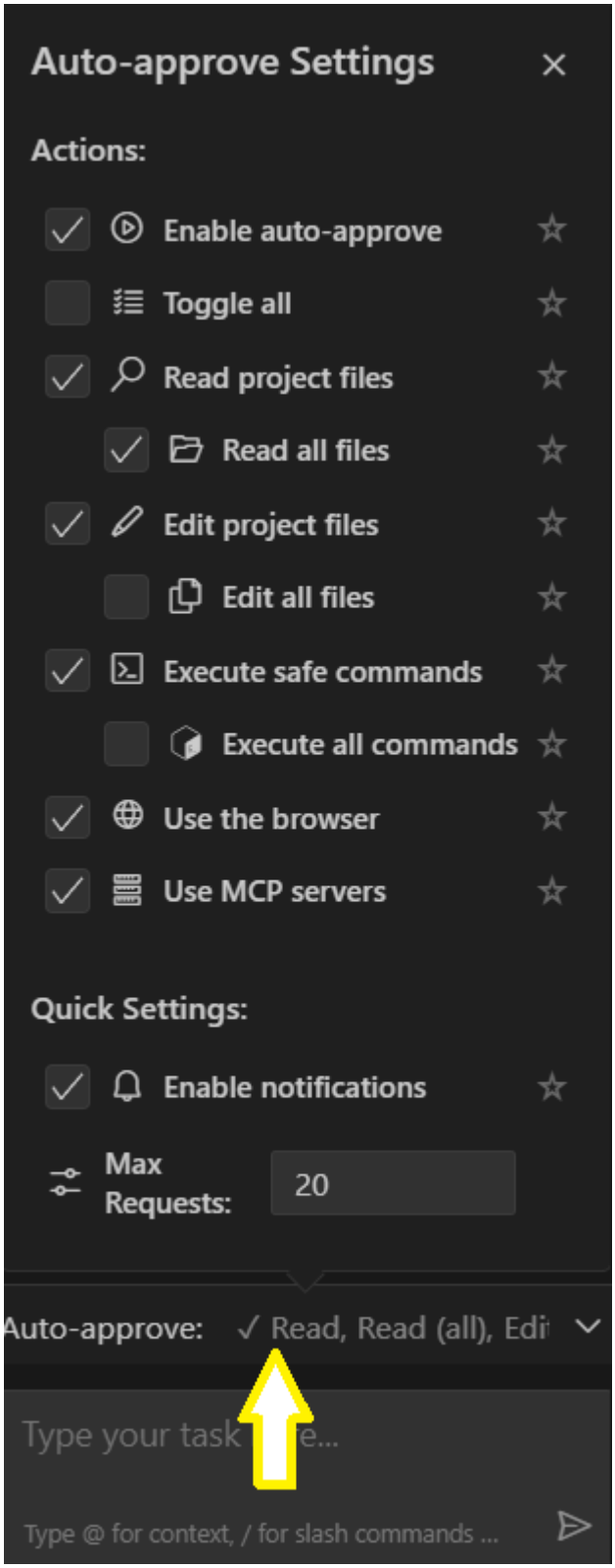
## How Cline Works?



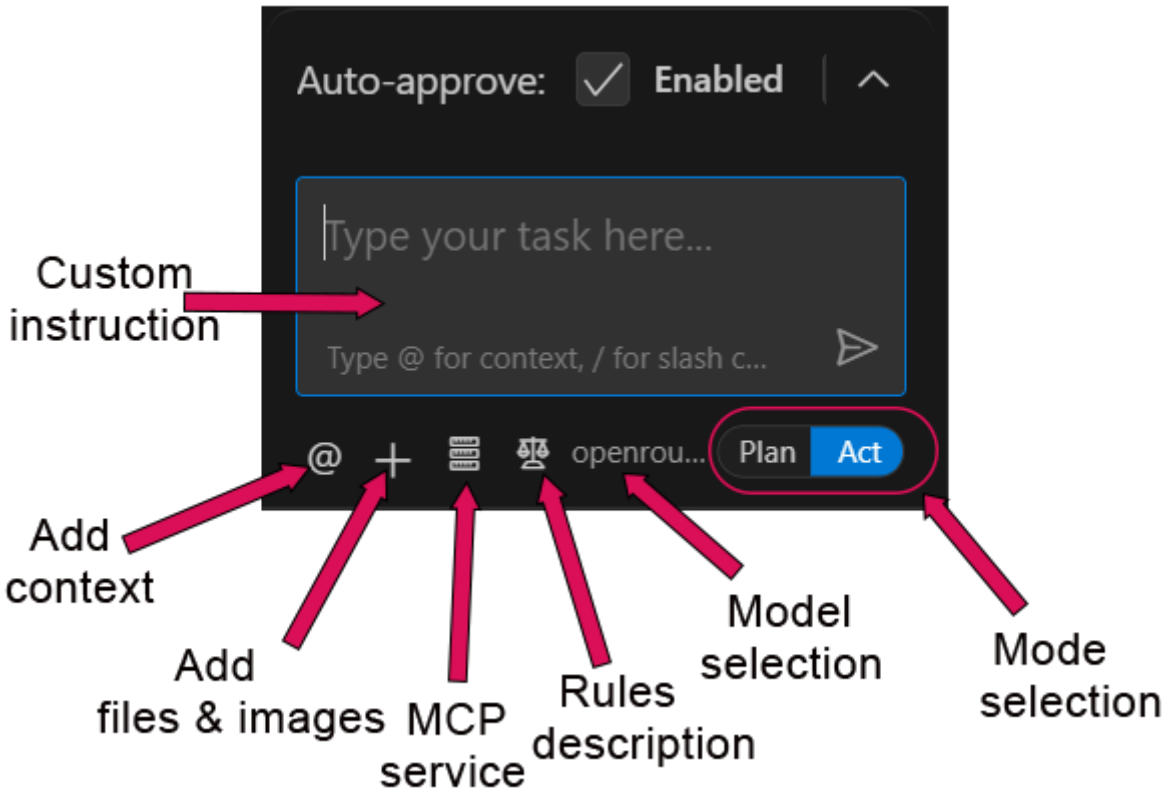

## Cline Basic Setup:

- Please follow the instructions specified in `Week0 -> Day2 -> Week 0- Day 2 - 02 VS code configurations.pdf` to install the Cline extension and configure model selection

- Click the `Auto-approve` field to set the cline permission

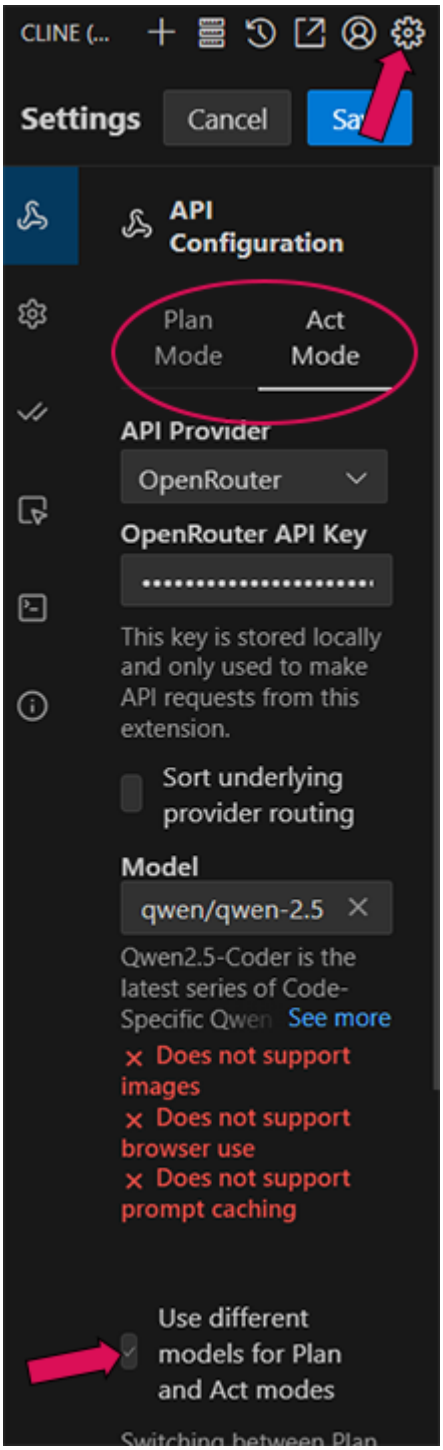


# Cline Basic Components and Their Descriptions:



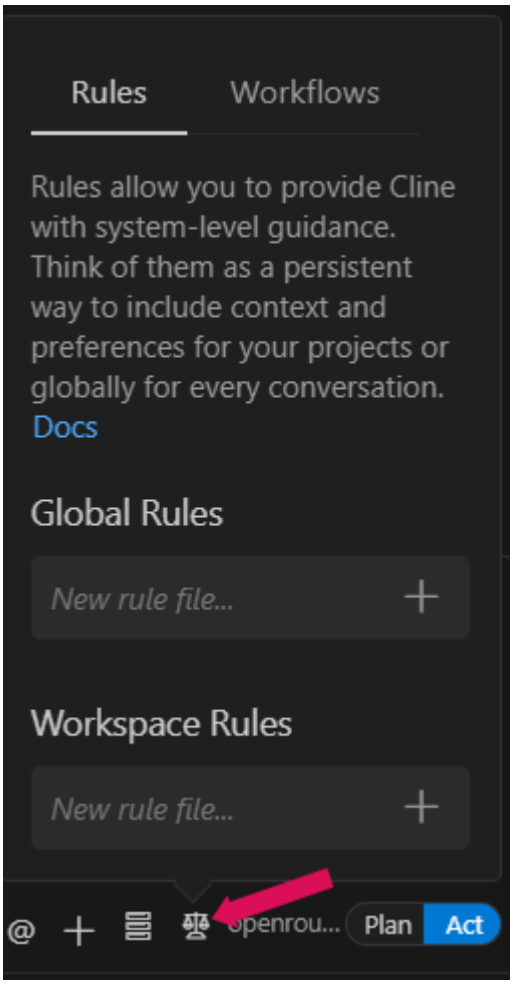

1. **Mode Selection:**
    - **Plan Mode:** This feature helps users interact with the Cline agent to create a clear plan or strategy for problem-solving and workflow creation. This mode is particularly helpful for establishing clear strategies and steps for code generation, resulting in more accurate code output.
    - **Act Mode:** Act mode implements the strategy created during Plan mode and generates deployable code. This separation allows for better quality control and more thoughtful implementation.
2. **Model Selection:**
    - Any compatible model can be selected after providing the necessary API key
    - Different models can be selected for Plan and Act modes, which can increase accuracy, performance, and control costs

- **Best Practice:** Use a reasoning-focused model for Plan mode where logical thinking is crucial, and a more cost-effective coding model for Act mode where code generation is the primary objective



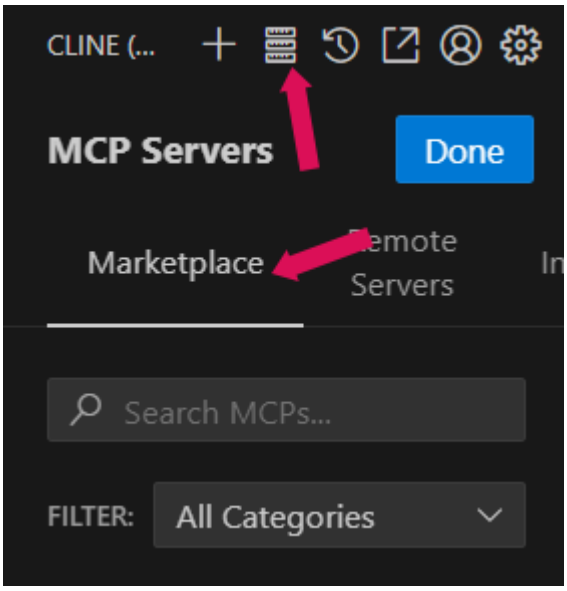

3. **Rules Configuration**
    - **Global Rules:** Set specific rules that apply to all projects using Cline. These rules ensure consistent behavior across different workspaces.
        - **Example Use Cases:**
            - Security rules preventing access to sensitive files
            - Coding standards and conventions
            - Prohibited operations or file types
        - **Setup:** Copy and paste the content from `Week1 -> Day3 -> global_rules.txt`

- **Workspace Rules:** Project-specific rules that apply only to the current workspace. These rules can be more granular and tailored to specific project requirements.
  - **Setup Methods:**
    - Click the `+` button in the workspace rules panel
    - or Create a `.clinerules` file in the project root directory
    - Use content from `Week1 -> Day3 -> project_rules.txt` to create the `.clinerules` file.
- **Cline Memory Bank:** The Memory Bank transforms Cline into a self-documenting development system that maintains context across sessions through structured documentation. This feature enables Cline to remember project details, coding patterns, and decisions made in previous sessions.
  - **Setup Process:**
    - Copy content from `Week1 -> Day3 -> cline_memorybank.txt`
    - Add the content as a rule in Cline
    - Initialize the memory bank by typing `Initialize mermory bank` in the Cline prompt

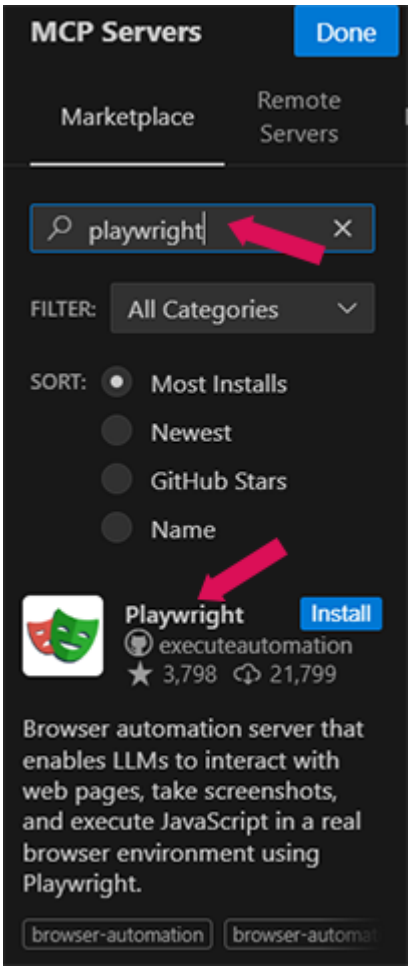


4. **Add Files and Images:**
    - Enhance Cline's understanding by providing additional context through:
        - **Documentation files:** README files, specifications, requirements
        - **Images:** UI mockups, flowcharts, architectural diagrams
        - **Configuration files:** Environment settings, build configurations
        - **Example code:** Reference implementations or templates
5. **Add Context:**
    - The `@` symbol enables context addition from specific sources:
        - **@filename:** Reference specific project files
        - **@terminal:** Include terminal output or commands
        - **@url:** Add context from web resources
        - **@folder:** Include entire directory contexts
    - **Best Practice:** Use context strategically to provide relevant information without overwhelming the model with unnecessary details.
6. **Slash Commands:**
    - **/smol Command:**
        - Condenses chat history within your current task. This is particularly useful when:
            - Conversations become very long
            - Performance starts to degrade
            - The model needs to refocus on core objectives
    - **/newtask Command:**
        - Creates a new task while preserving context from your current session.
            - **Usage Examples:**
                - `/newtask` Start a completely new task
                - `/newtask add this as a new feature` Branch into a new feature while maintaining context
            - This is perfect for:
                - Branching explorations
                - Starting fresh without losing progress
                - Organizing complex projects into manageable tasks
7. **MCP Services:**
    - The Model Context Protocol is an open standard that enables developers to build secure, two-way connections between their data sources and AI-powered tools. Think of MCP as creating a universal language that allows AI systems like Cline to communicate seamlessly with external services, databases, and tools that data scientists use daily.

- **Installation Process:**
  - **Access MCP Marketplace:** Cline provides automated installation, configuration management, and orchestration of AWS Labs servers with centralized logging and environment control
  - **Service Configuration:** Configure each MCP server with appropriate credentials and connection parameters
  - **Testing Connections:** Verify that Cline can successfully communicate with your chosen MCP services
  - **Workflow Integration:** Incorporate MCP services into your data science workflows
- **Security Considerations:**
  - Store API keys and credentials securely
  - Use environment variables for sensitive information
  - Implement proper access controls
  - Regular credential rotation
- **Playwright MCP Service Installation:**
  1. First click on the `MCP servers` icon located on the top right corner in the cline dashboard
  2. Then click on the Marketplace tab



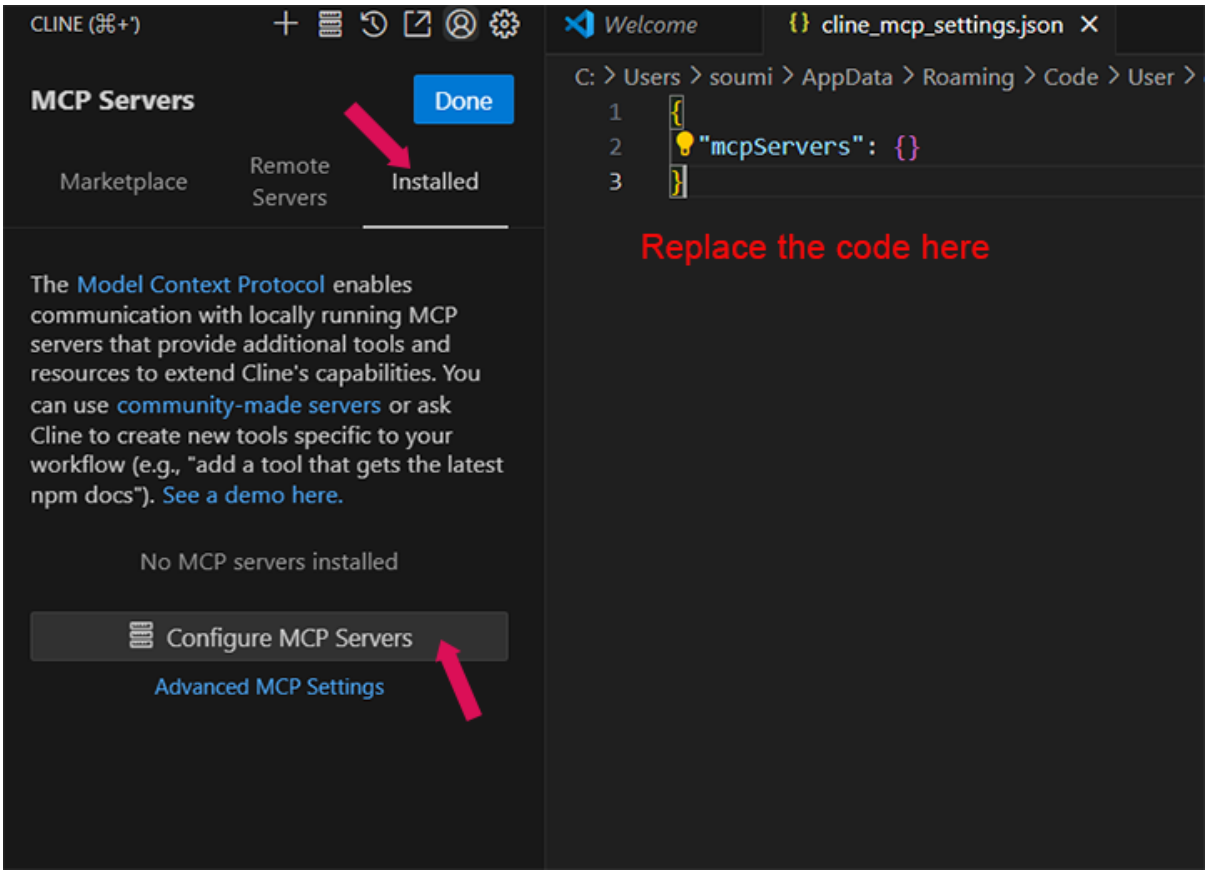

  3. Then inside the search bar type `playwright`
  4. You can directly install playwright by clicking on the `Install` button. This process requires LLMs support to install paywright
  5. Or you can install it manually. Click on the `playwright` icon. This will redirect you to the official github page.




6. You have to copy the code snippet present under `Configuration to use Playwright Server` heading.

```
{
   "mcpServers": {
      "playwright": {
         "command": "npx",
         "args": ["-y", "@executeautomation/playwright-mcp-server"]
      }
   }
}
```

7. Return to cline and click on the `Install` tab.
8. Then click on the `Configure MCP Servers` ribbon that will open a `.json` file in the workspace
9. Paste the previous copied code snippet in the opened `.json` file and save it.




10. According to your computer configuration you may find the following error `npx' is not recognized as an internal or external command`. In that case you have to install `Node.js` library.

11. Go to the following link `https://nodejs.org/en` and download the latest LTS version of the Node.js for windows.

12. Now run the istaller and follow the installaton wizard. Make sure to chek `ADD TO PATH` option during installation.

13. Now install playwright package & browser to your current environment. Run the followign commands `npm install -g playwright` and `npx playwright install` one by ones in your console environment.

14. Then click on `Done` to complete the process