

Thomas Bley

Your Code Runs in Docker, Why Not Your IDE?

PHP UG Darmstadt
October 2021

Bringmeister.de
Dein Online Supermarkt

About me

- Senior PHP Developer
- Linux, PHP, MySQL since 2001
- studied at TU München
- working for Bringmeister in Berlin



Why use Docker?

- easy to manage different isolated environments
- quick and easy to setup development environments
- easy to share with other developers
- easy to integrate in continuous integrations
- easy to standardize environments
- easy to test new things
- write once, run anywhere
- big ecosystem with Docker Hub

→ we love to run our code in Docker

Why use an IDE?

IDE = Integrated Development Environment

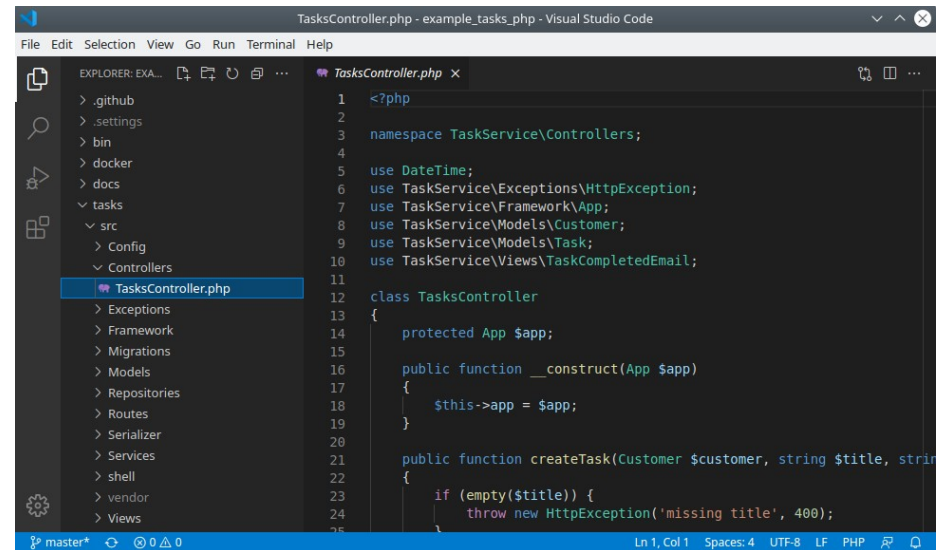
- flexible text editor
- code completion, formatting
- automatic checks for errors, debugging
- integrated version control
- many other features with extensions

→ more productivity, more quality

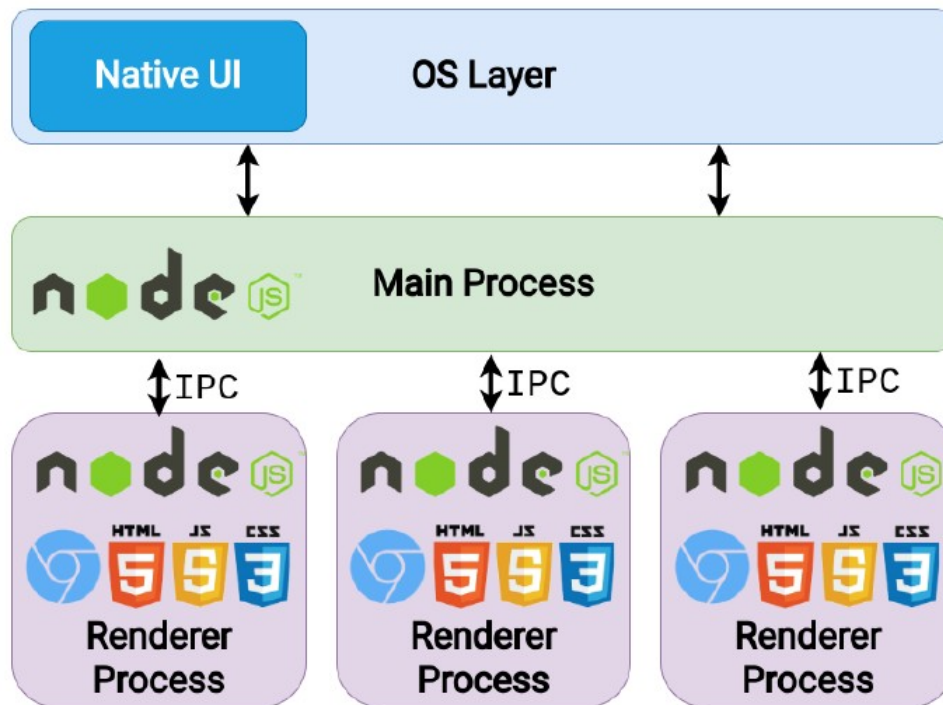
but: installed as desktop application, configured manually

What is Visual Studio Code?

- free and open source IDE from Microsoft
- lightweight, very fast
- supports many programming languages
- big community
- many extensions
- desktop application, based on Electron (by GitHub)
- marketplace is not open source, proprietary license



What is Electron?



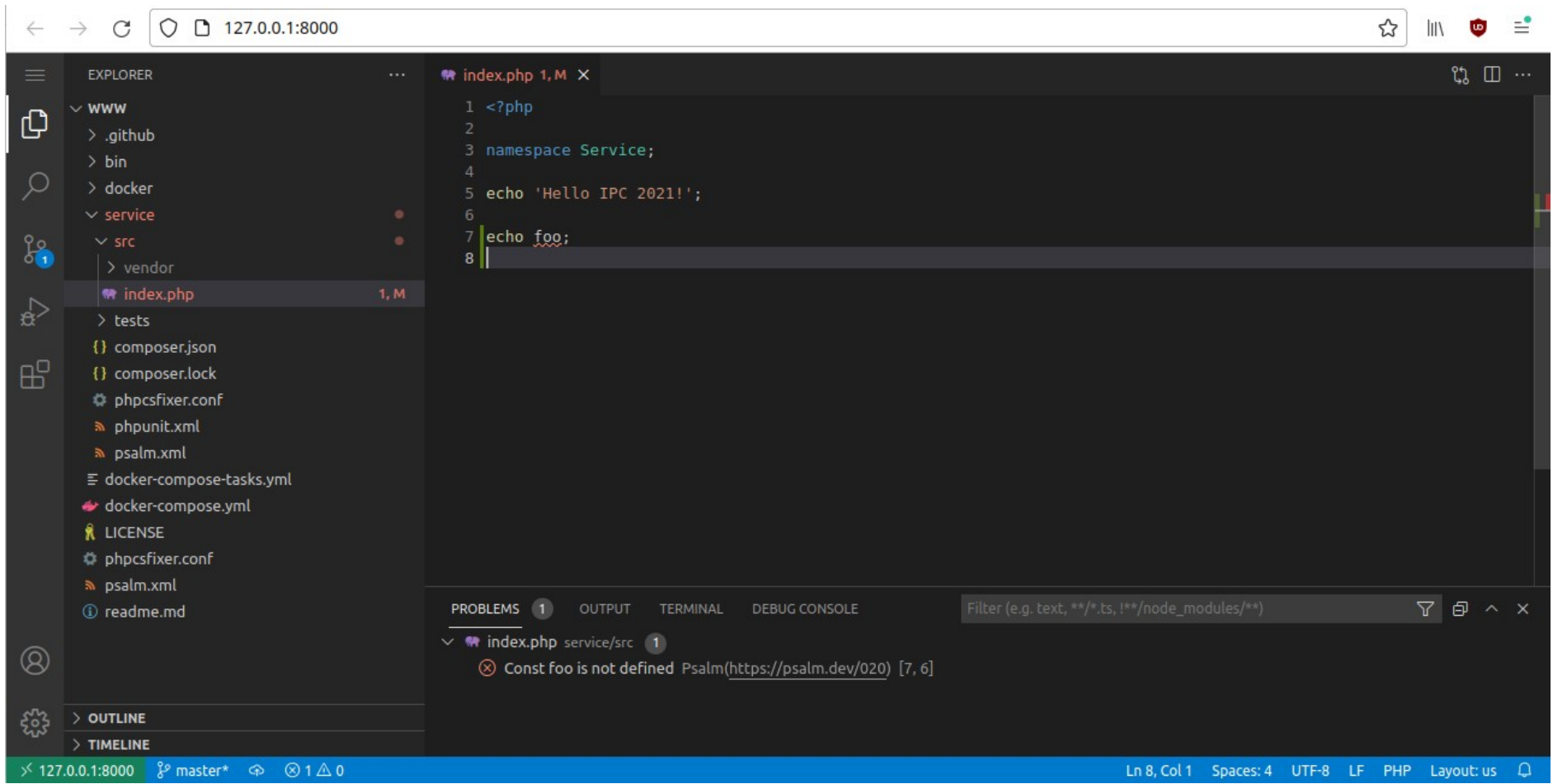
Framework for cross platform desktop applications from GitHub

- combines the Chromium rendering engine (Frontend)
- with node.js runtime (Backend)

source:

livecodestream.dev/post/how-to-build-desktop-applications-using-electron-the-right-way/

Can we run VS Code in the browser?



Code-Server

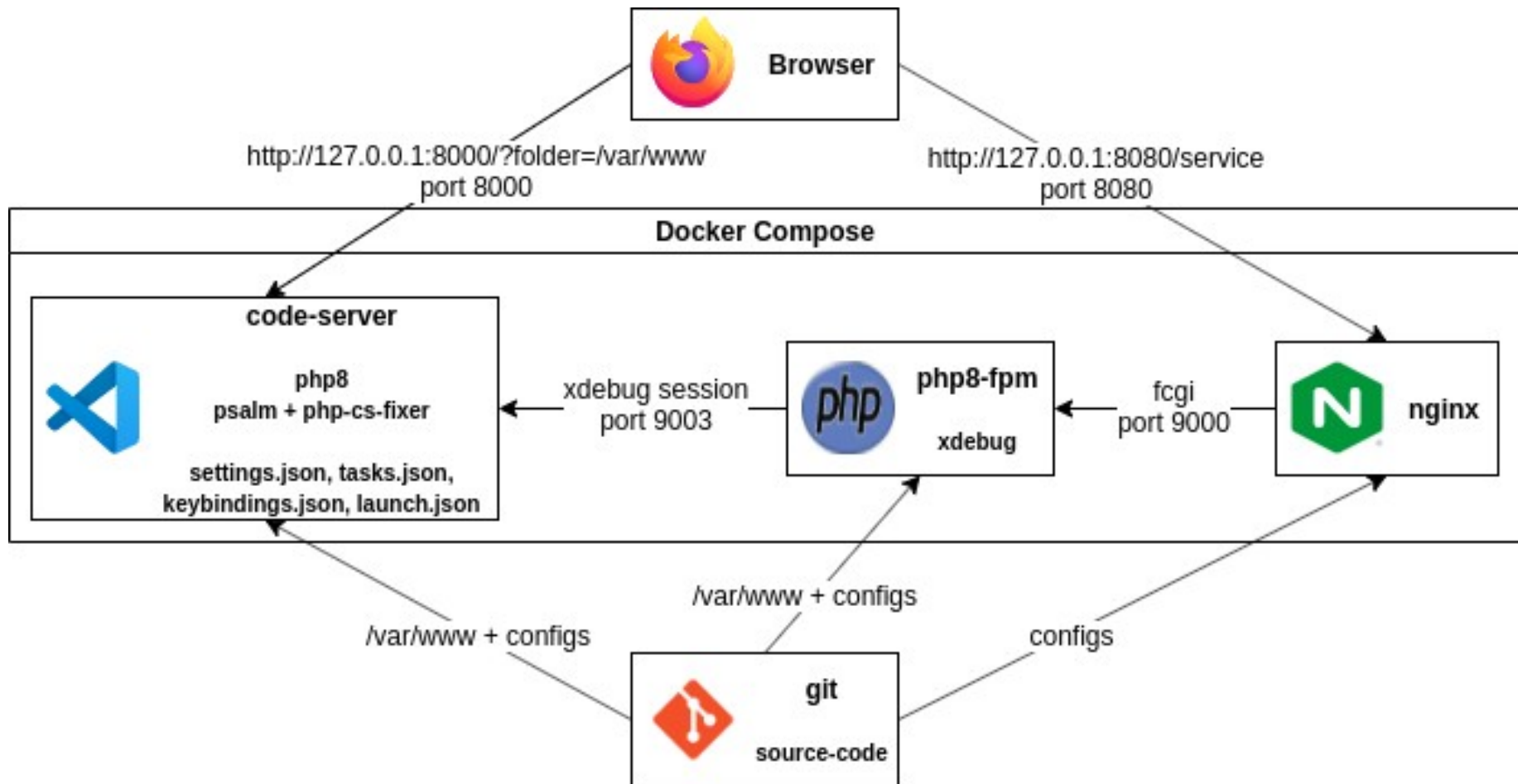
- “VS Code in the browser”
- fork of VS Code, open source
- runs backend part on server, frontend part in browser
- supports VSX extensions
- can be installed as Debian package (95M)
- not allowed to use VS Code Marketplace
- uses Open VSX Registry as marketplace (open-vsx.org)

sources:

github.com/cdr/code-server

github.com/cdr/code-server/blob/main/docs/FAQ.md

Setup



Dockerfile

```
# Setup a Debian container with PHP 8.0 and code-server
```

```
# docker-compose build --build-arg UID=$(id -u)
```

```
FROM debian:bullseye-slim
```

```
RUN apt-get -y update \
```

```
&& DEBIAN_FRONTEND=noninteractive apt-get -y --no-install-recommends install apt-transport-https curl \
    ca-certificates \
```

```
&& echo "deb https://packages.sury.org/php/ bullseye main" >/etc/apt/sources.list.d/ondrej-debian-php.list \
```

```
&& curl -s https://packages.sury.org/php/apt.gpg >/etc/apt/trusted.gpg.d/php.gpg \
```

```
&& apt-get -y update \
```

```
&& DEBIAN_FRONTEND=noninteractive apt-get -y --no-install-recommends install git php8.0-cli \
    php8.0-mbstring php8.0-curl php8.0-xml php8.0-zip \
```

```
&& curl -#L -o /tmp/code-server.deb \
```

```
https://github.com/cdr/code-server/releases/download/v3.12.0/code-server_3.12.0_amd64.deb \
```

```
&& dpkg -i /tmp/code-server.deb
```

```
ARG UID
```

```
RUN useradd -m docker -u ${UID} -d /config -s /bin/bash
```

```
USER docker
```

Dockerfile #2

```
# Install VS Code extensions, start code-server
```

```
ENV SERVICE_URL=https://open-vsx.org/vscode/gallery
```

```
ENV ITEM_URL=https://open-vsx.org/vscode/item
```

```
ADD LouisWT.regex-preview-0.1.5.vsix /tmp/
```

```
RUN code-server --install-extension felixfbecker.php-debug \  
    --install-extension fterrag.vscode-php-cs-fixer \  
    --install-extension alphabotsec.vscode-eclipse-keybindings \  
    --install-extension hediet.vscode-drawio \  
    --install-extension getpsalm.psalms-vscode-plugin \  
    --install-extension /tmp/LouisWT.regex-preview-0.1.5.vsix \  
    && chmod -R a+w /config
```

```
WORKDIR /var/www
```

```
EXPOSE 8000
```

```
CMD [ "code-server", "--auth", "none", "--disable-telemetry", "--disable-update-check", "--bind-addr", "0.0.0.0:8000" ]
```

docker-compose.yml

code-server:

build:

context: ./docker/code-server

args: **[UID]**

cap_drop: [all]

userns_mode: "host"

volumes:

- **./:/var/www**
- ./bin:/usr/local/sbin
- ./docker/code-server/**settings.json**:/config/.local/share/code-server/User/settings.json
- ./docker/code-server/**tasks.json**:/config/.local/share/code-server/User/tasks.json
- ./docker/code-server/**keybindings.json**:/config/.local/share/code-server/User/keybindings.json
- ./docker/code-server/**launch.json**:/var/www/.vscode/launch.json #xdebug

ports:

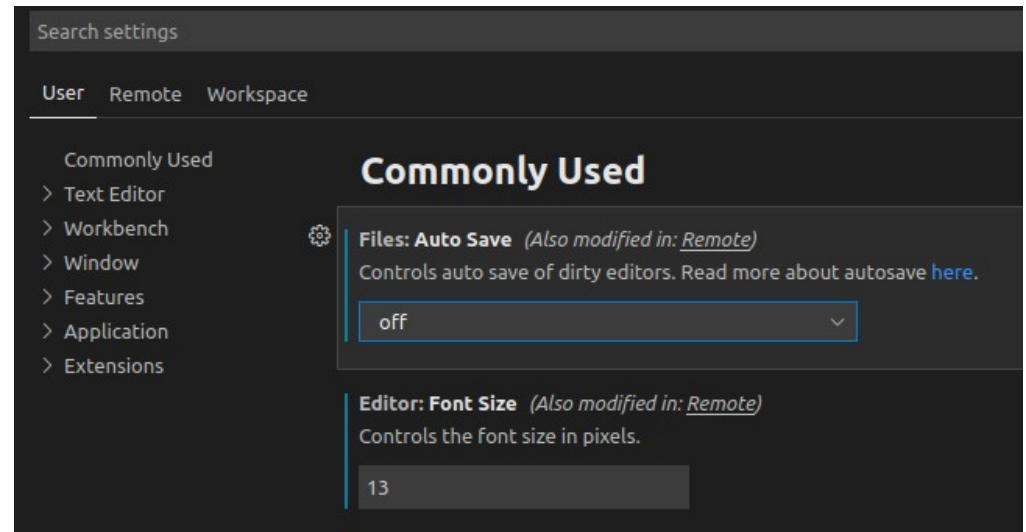
- "**127.0.0.1:8000**:8000"

Pre-Configure VS Code settings

settings.json

```
{  
  "files.autoSave": "afterDelay",  
  "editor.minimap.enabled": false,  
  "editor.renderIndentGuides": false,  
  "editor.fontSize": 13,  
  "workbench.colorTheme": "Default Dark+",  
  ....  
}
```

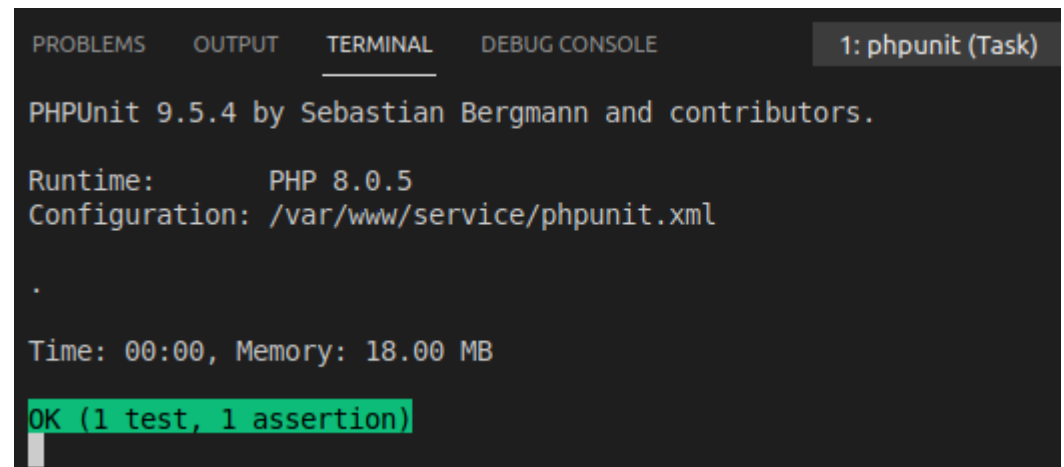
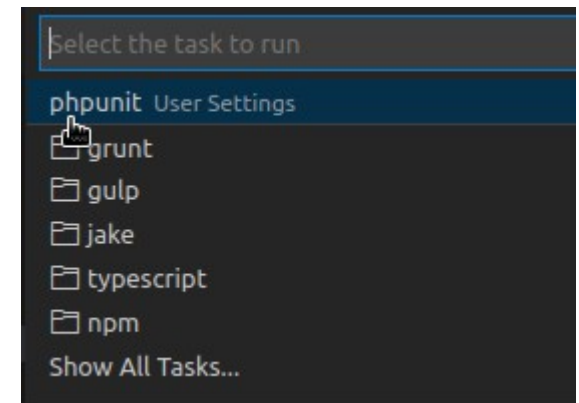
```
git diff docker/code-server/settings.json  
+++ b/docker/code-server/settings.json  
@@ -35,5 +35,6 @@  
+   "files.autoSave": "off"  
}
```



Can we run tasks?

tasks.json (Ctrl + Shift + t)

```
{
  // See https://code.visualstudio.com/docs/editor/tasks#vscode
  "version": "2.0.0",
  "tasks": [{
    "label": "phpunit",
    "type": "shell",
    "problemMatcher": [],
    "command": "phpunit.phar",
    "options": {"cwd": "/var/www/service"},
    "presentation": {
      "showReuseMessage": false,
      "echo": false,
      "clear": true
    }
  }]
}
```



How about keyboard shortcuts?

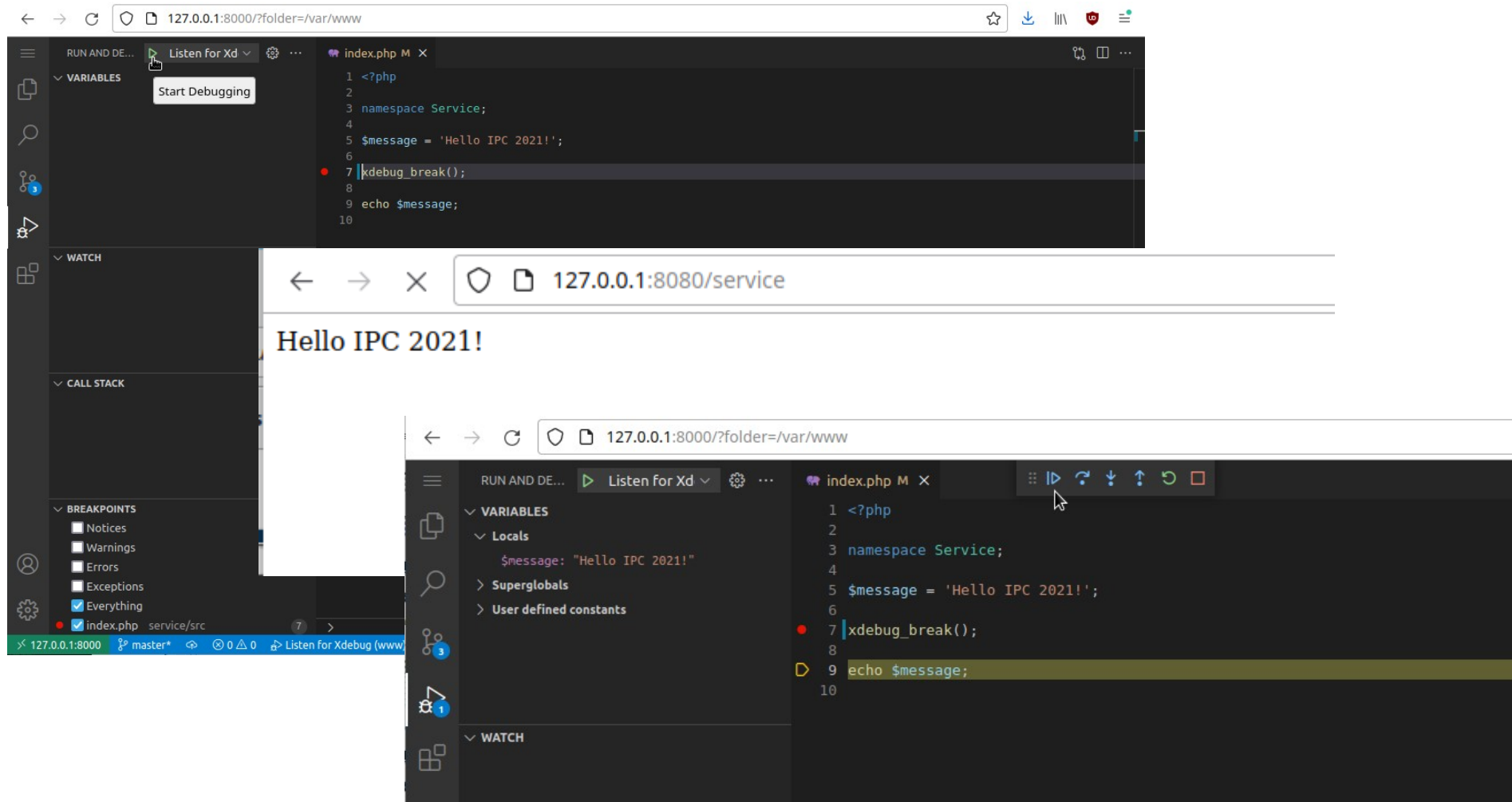
```
# keybindings.json
```

```
[  
  {"key": "ctrl+shift+t", "command": "workbench.action.tasks.runTask"},  
  
  // switch file tabs, normally: ctrl+tab  
  {"key": "ctrl+[Backquote]", "command":  
    "workbench.action.quickOpenLeastRecentlyUsedEditorInGroup"},  
  {"key": "ctrl+[Backquote]", "command":  
    "workbench.action.quickOpenNavigateNextInEditorPicker",  
    "when": "inEditorsPicker && inQuickOpen"},  
  
  // close active file tab, normally: ctrl+w  
  {"key": "ctrl+e", "command": "workbench.action.closeActiveEditor"}  
]
```

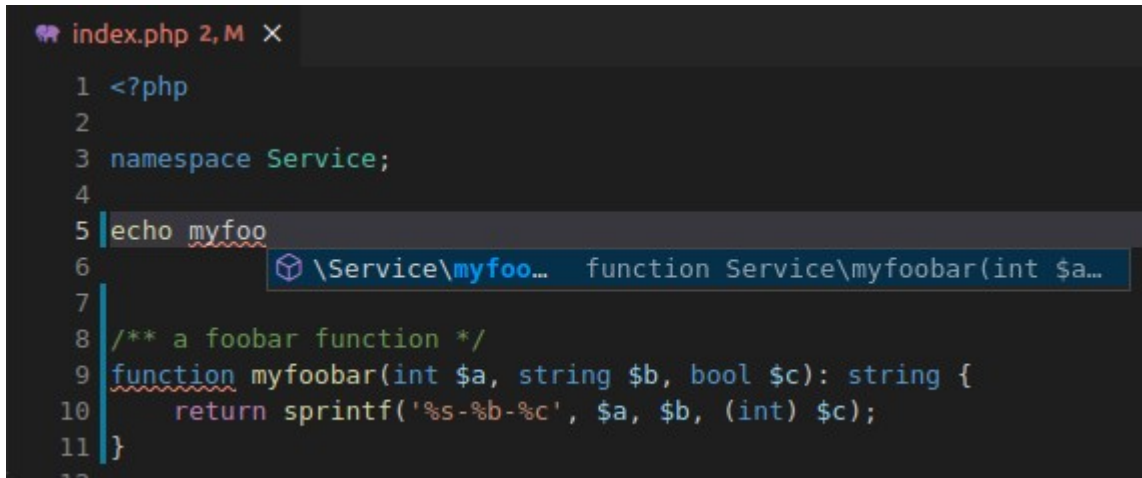
```
# Some keyboard shortcuts are exclusively reserved to the browser (e.g. ctrl+tab).
```

```
# With Chrome you can use "Menu -> Install code-server" to run code-server  
as a Chrome App to bypass reserved keyboard shortcuts.
```

Debugging (xdebug)

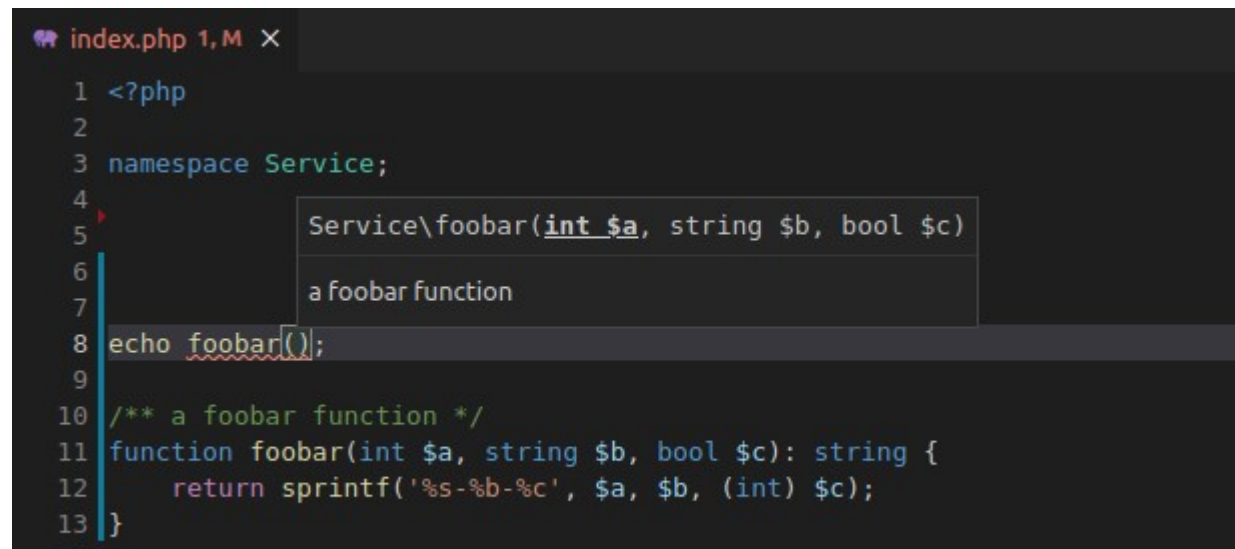


Auto-Completion (psalm language server)



A screenshot of a code editor window titled 'index.php 2, M'. The code is in PHP and shows a namespace 'Service' and a function 'myfoobar'. The cursor is on line 5, typing 'echo myfoo'. A dropdown menu shows the suggestion '\Service\myfoo...' followed by 'function Service\myfoobar(int \$a...'. The code is as follows:

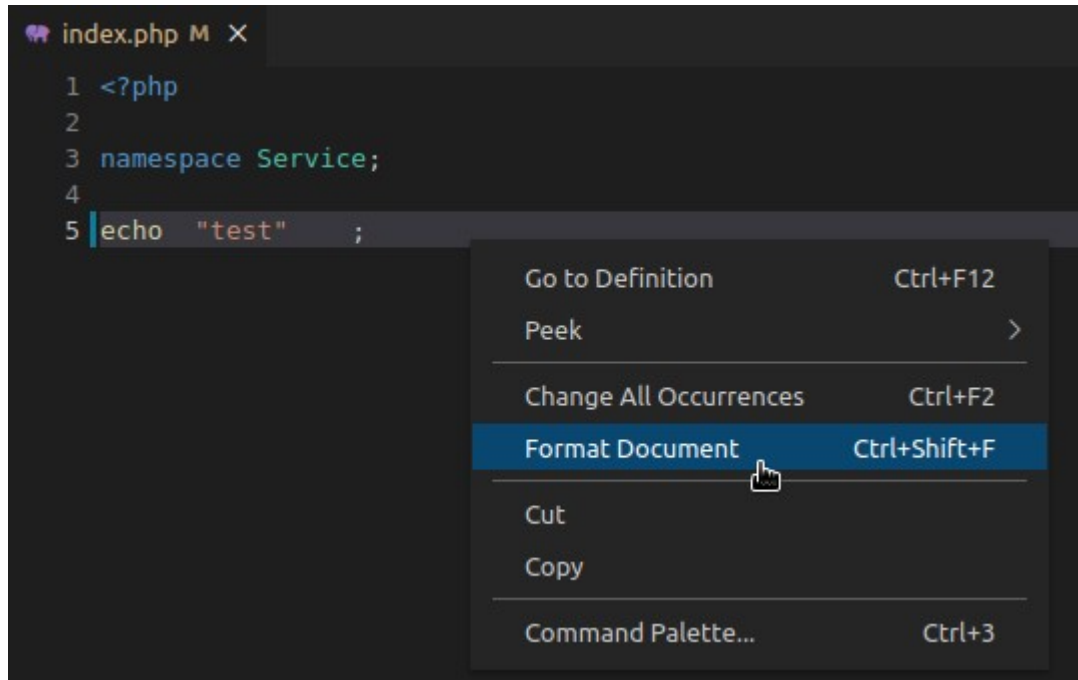
```
1 <?php
2
3 namespace Service;
4
5 echo myfoo
6
7
8 /** a foobar function */
9 function myfoobar(int $a, string $b, bool $c): string {
10     return sprintf('%s-%b-%c', $a, $b, (int) $c);
11 }
12
```



A screenshot of a code editor window titled 'index.php 1, M'. The code is in PHP and shows a namespace 'Service' and a function 'foobar'. The cursor is on line 8, typing 'echo foobar()'. A dropdown menu shows two suggestions: 'Service\foobar(int \$a, string \$b, bool \$c)' and 'a foobar function'. The code is as follows:

```
1 <?php
2
3 namespace Service;
4
5
6
7
8 echo foobar();
9
10 /** a foobar function */
11 function foobar(int $a, string $b, bool $c): string {
12     return sprintf('%s-%b-%c', $a, $b, (int) $c);
13 }
14
```

Auto-Formatting (php-cs-fixer)

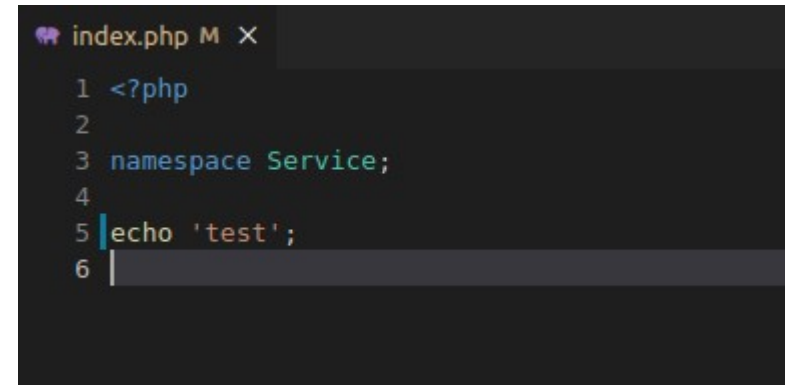


A screenshot of an IDE window titled 'index.php M X'. The code in the editor is:

```
1 <?php
2
3 namespace Service;
4
5 echo "test" ;
```

A context menu is open over the code, with the following items:

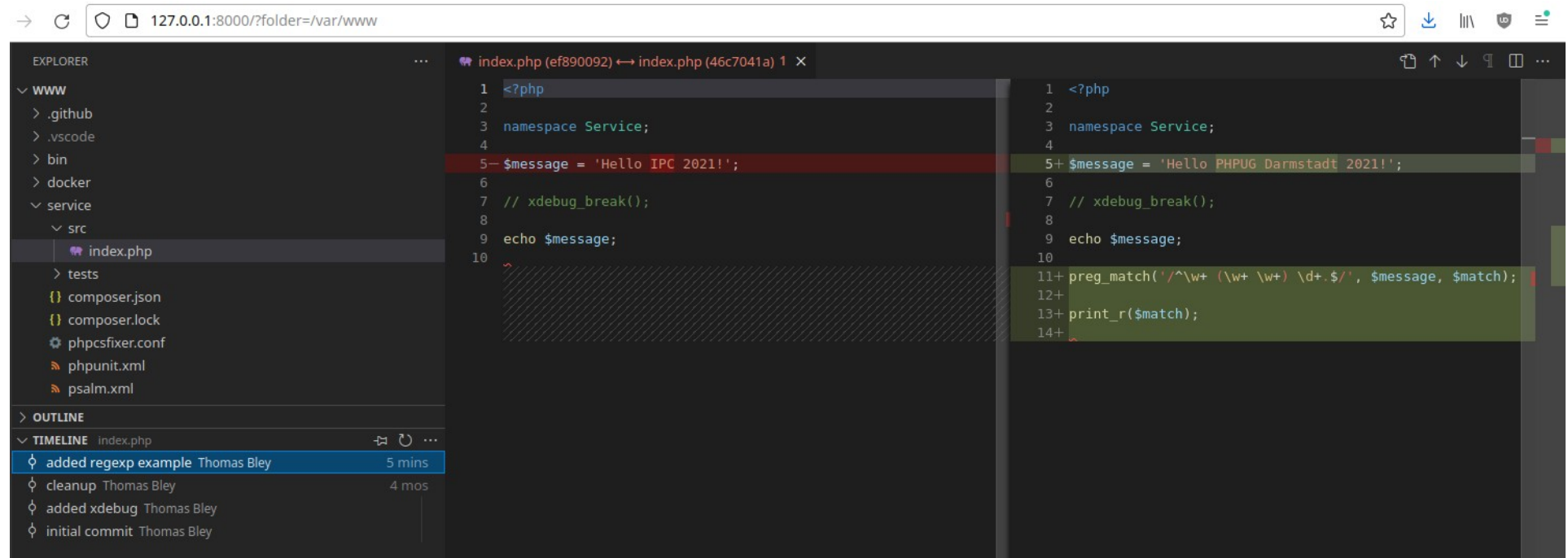
- Go to Definition (Ctrl+F12)
- Peek (>)
- Change All Occurrences (Ctrl+F2)
- Format Document (Ctrl+Shift+F)** (highlighted with a mouse cursor)
- Cut
- Copy
- Command Palette... (Ctrl+3)



The same IDE window after the 'Format Document' action. The code is now formatted with single quotes and no trailing space:

```
1 <?php
2
3 namespace Service;
4
5 echo 'test';
6
```

Git



Regex preview

The screenshot shows an IDE with a PHP file named `index.php` and a side panel titled `Explain RegExp`. The PHP code is as follows:

```
1 <?php
2
3 namespace Service;
4
5 $message = 'Hello PHPUG Darmstadt 2021!';
6
7 // xdebug_break();
8
9 echo $message;
10
11 preg_match('/^\w+ (\w+ \w+) \d+.$/', $message, $match);
12
13 print_r($match);
14
```

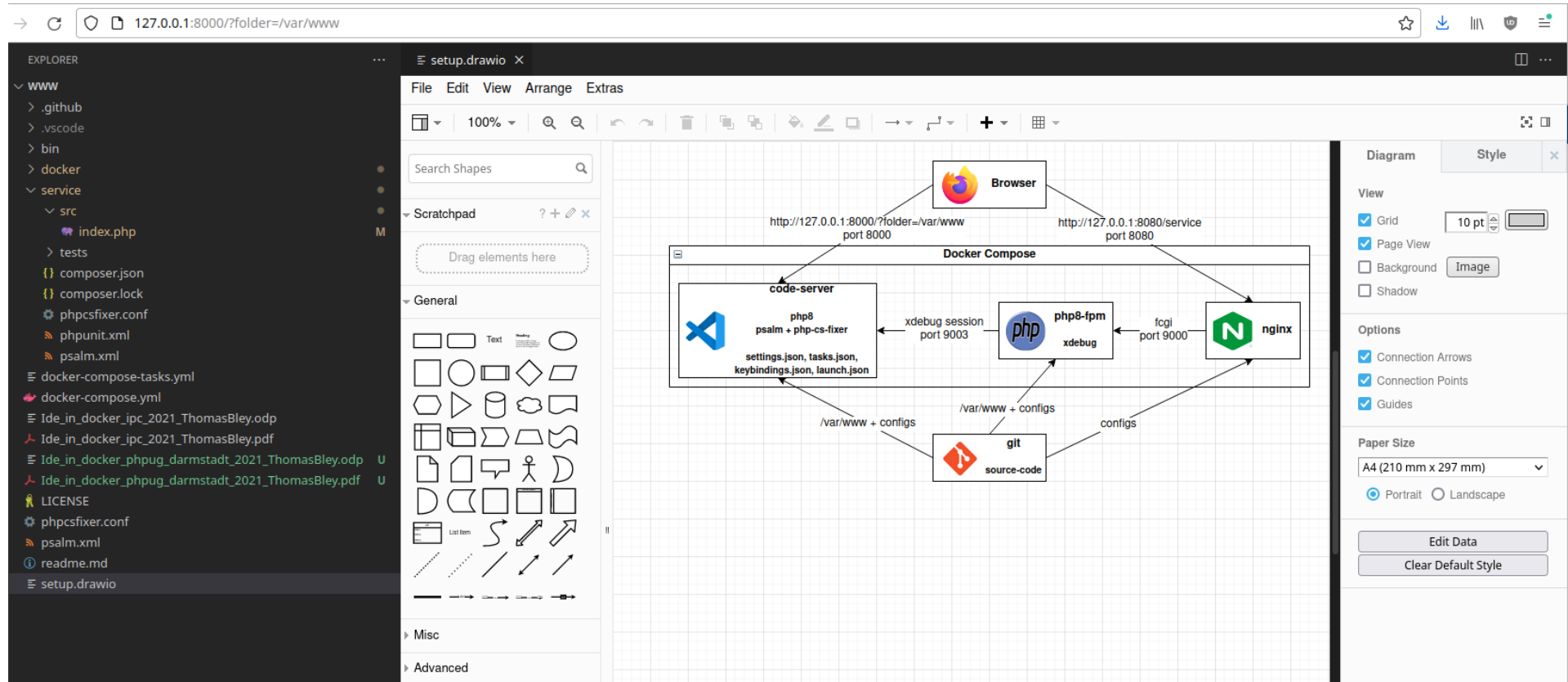
The `Explain RegExp` panel displays the regex pattern `^\w+ (\w+ \w+) \d+.$` and a corresponding state transition diagram. The diagram illustrates the following components:

- Start of line**: The starting point of the regex engine.
- word**: A sequence of word characters, represented by a green box.
- group #1**: A capturing group containing two words separated by a space, represented by a dashed box around the second and third `word` boxes.
- digit**: A sequence of digits, represented by a green box.
- any character**: A single character, represented by a green box.
- End of line**: The final point of the regex engine.

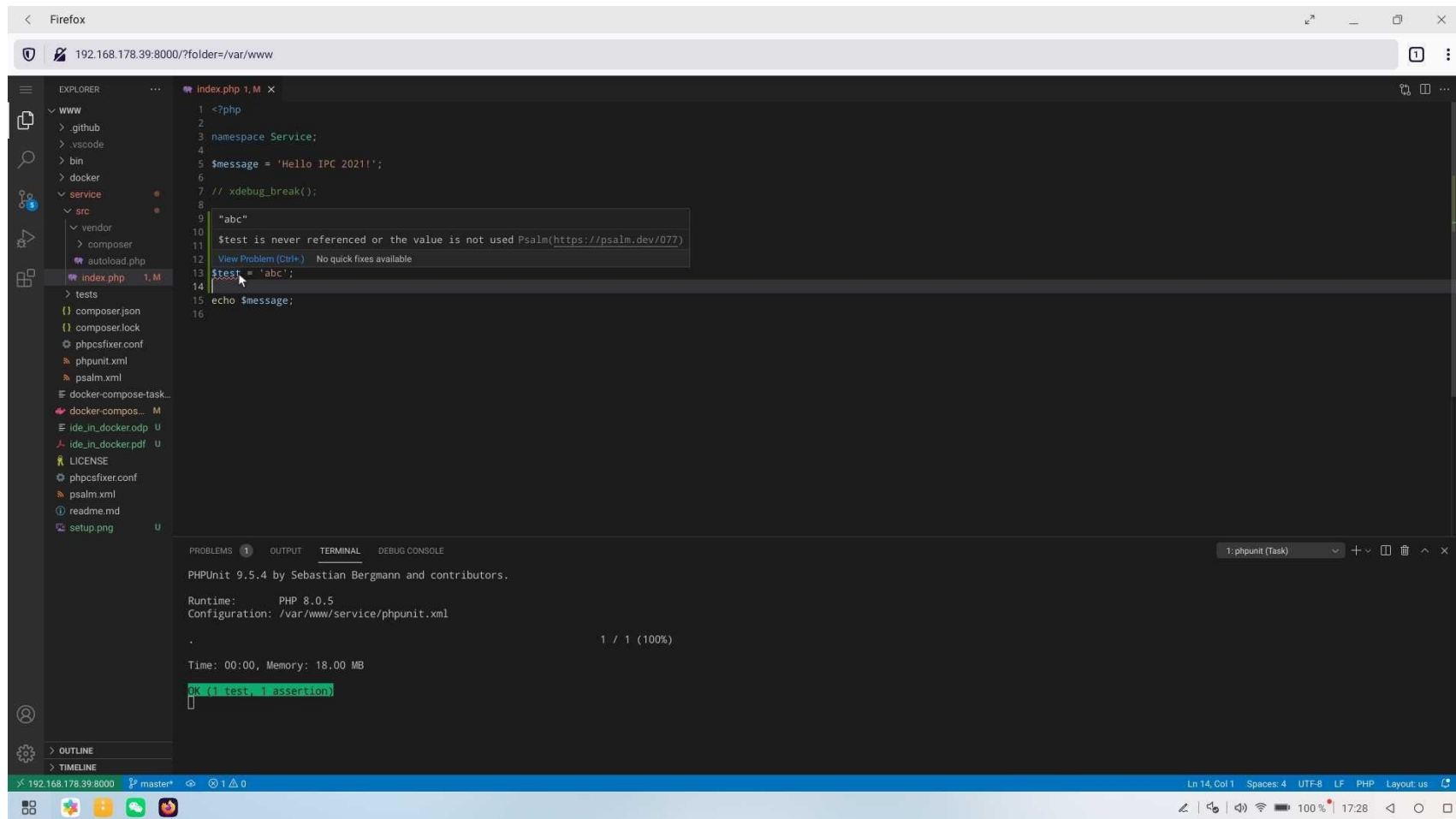
A context menu is open over the code, showing the following options:

- RegExp Explain
- Change All Occurrences `Ctrl+F2`
- Format Document `Ctrl+Shift+F`
- Cut
- Copy
- Command Palette... `Ctrl+3`

Draw.io



DEMO



Firefox 88 on Huawei P40 Desktop mode

Thanks for listening!

Questions?

slides and sources:

github.com/thomasbley/ide_in_docker

follow me:

twitter.com/thbley

