Introduction to GitHub Codespaces

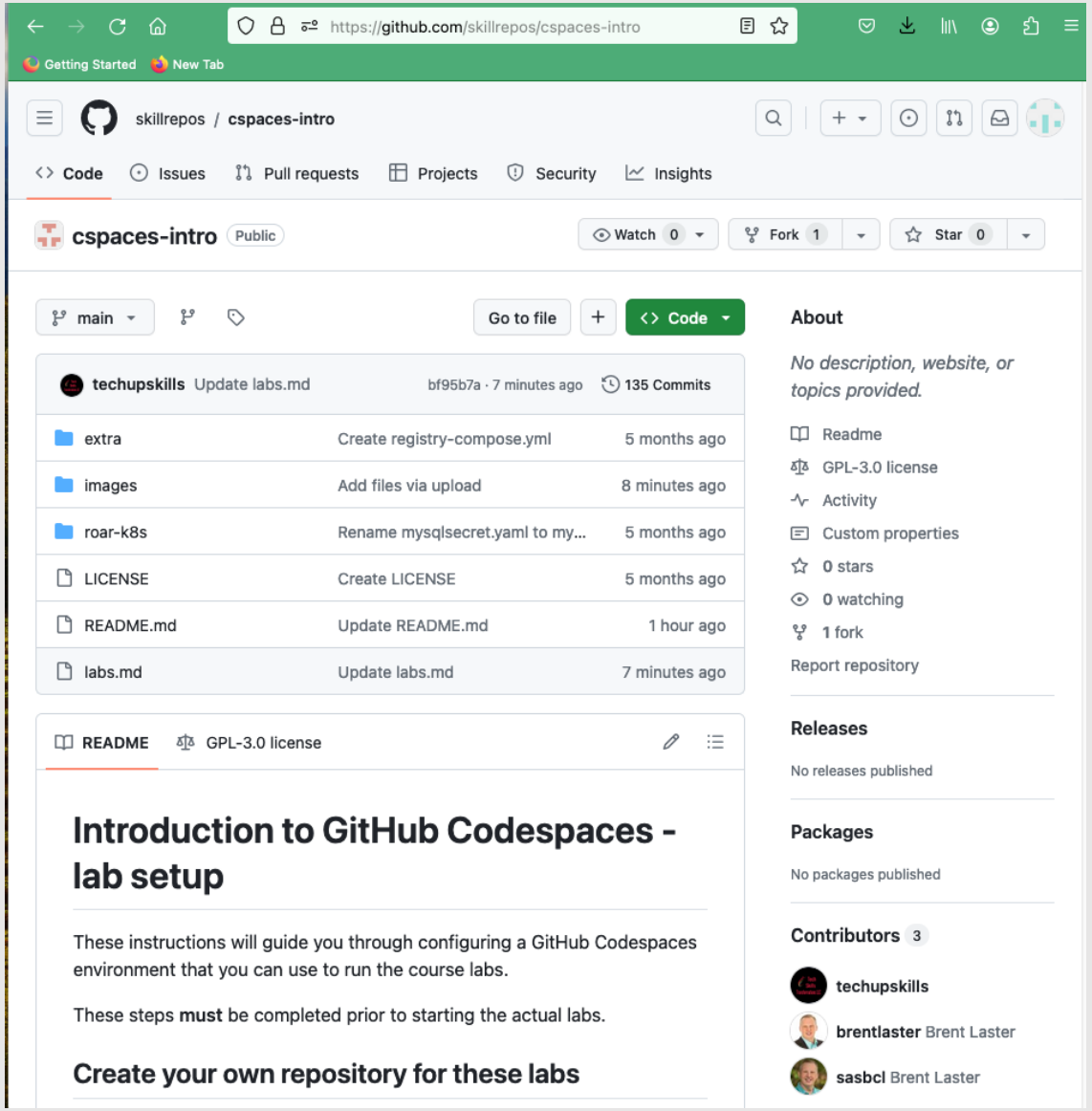
Presented by

Tech Skills Transformations LLC

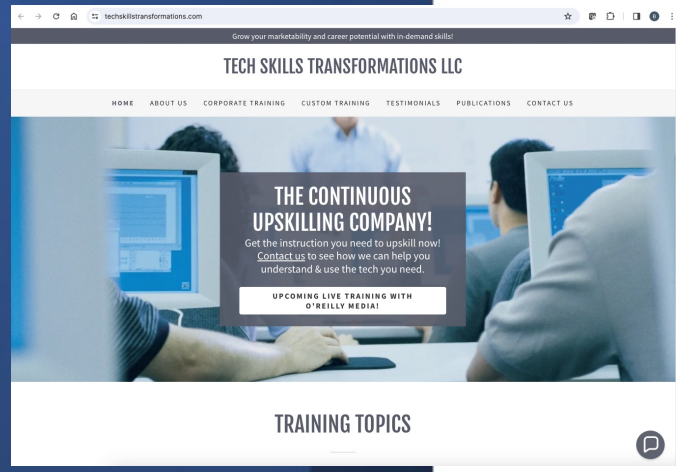
TECHUPSKILLS ®

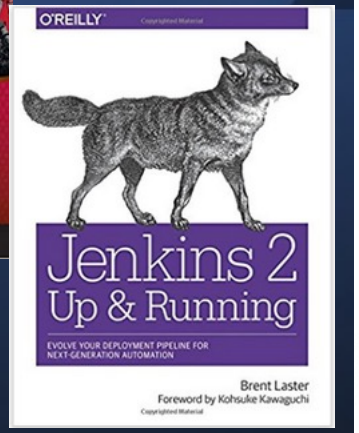
Welcome to the Introduction to GitHub Codespaces Workshop!

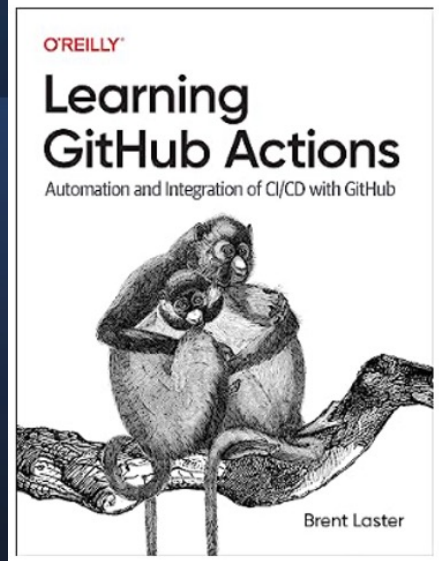
- Log into GitHub
- Go to https://github.com/skillrepos/cspaces-intro
- Fork project
- Follow instructions in README to setup codespace
- When done open *labs.md* file **with Preview**
- Keep copy of labs.md open in separate browser tab/window



About me







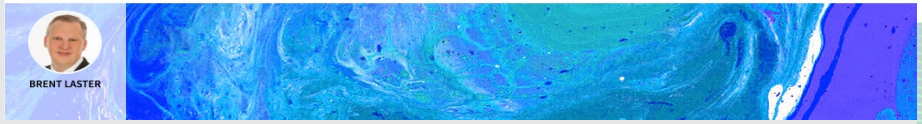
- Founder, Tech Skills Transformations LLC
- Global trainer – O'Reilly, NFJS, etc.
  - Git, GitHub, GitHub Actions
  - Jenkins, Gradle
  - Kubernetes, Tekton, Kustomize
  - Copilot, Codespaces
  - Etc.
- Author -
  - O'Reilly "reports"
  - Professional Git book
  - Jenkins 2 – Up and Running book
  - Learning GitHub Actions
- https://www.linkedin.com/in/brentlaster
- @BrentCLaster
- GitHub: brentlaster

O'Reilly Training

LIVE ONLINE TRAINING
Containers A-Z
An overview of containers, Docker, Kubernetes, Istio, Helm, Kubernetes Operators, and GitOps
Topic: System Administration

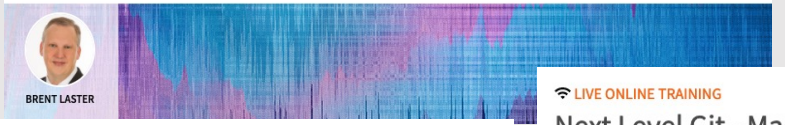

LIVE ONLINE TRAINING
Helm Fundamentals
Deploying, upgrading, and rolli
Topic: System Administration


LIVE ONLINE TRAINING
Git Fundamentals
Simplify and speed up management of your source code
Topic: Software Development


LIVE ONLINE TRAINING
Building a Kubernetes Operator: Extending Kubernetes to Fit Your Applications
Extending Kubernetes to Fit Your Applications
Topic: System Administration


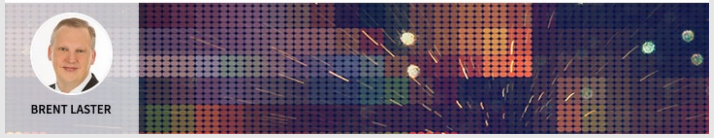
LIVE ONLINE TRAINING
Continuous Delivery in Kubernetes with ArgoCD
Automating deployment and lifecycle management
Topic: System Administration


LIVE ONLINE TRAINING
Getting started with continuous delivery (CD)
Move beyond CI to build, manage, and deploy a working pipeline
Topic: System Administration


LIVE ONLINE TRAINING
Next Level Git - Master your content
Use powerful tools in Git to simplify merges, rewrite history, and perform automatic updates
Topic: Software Development


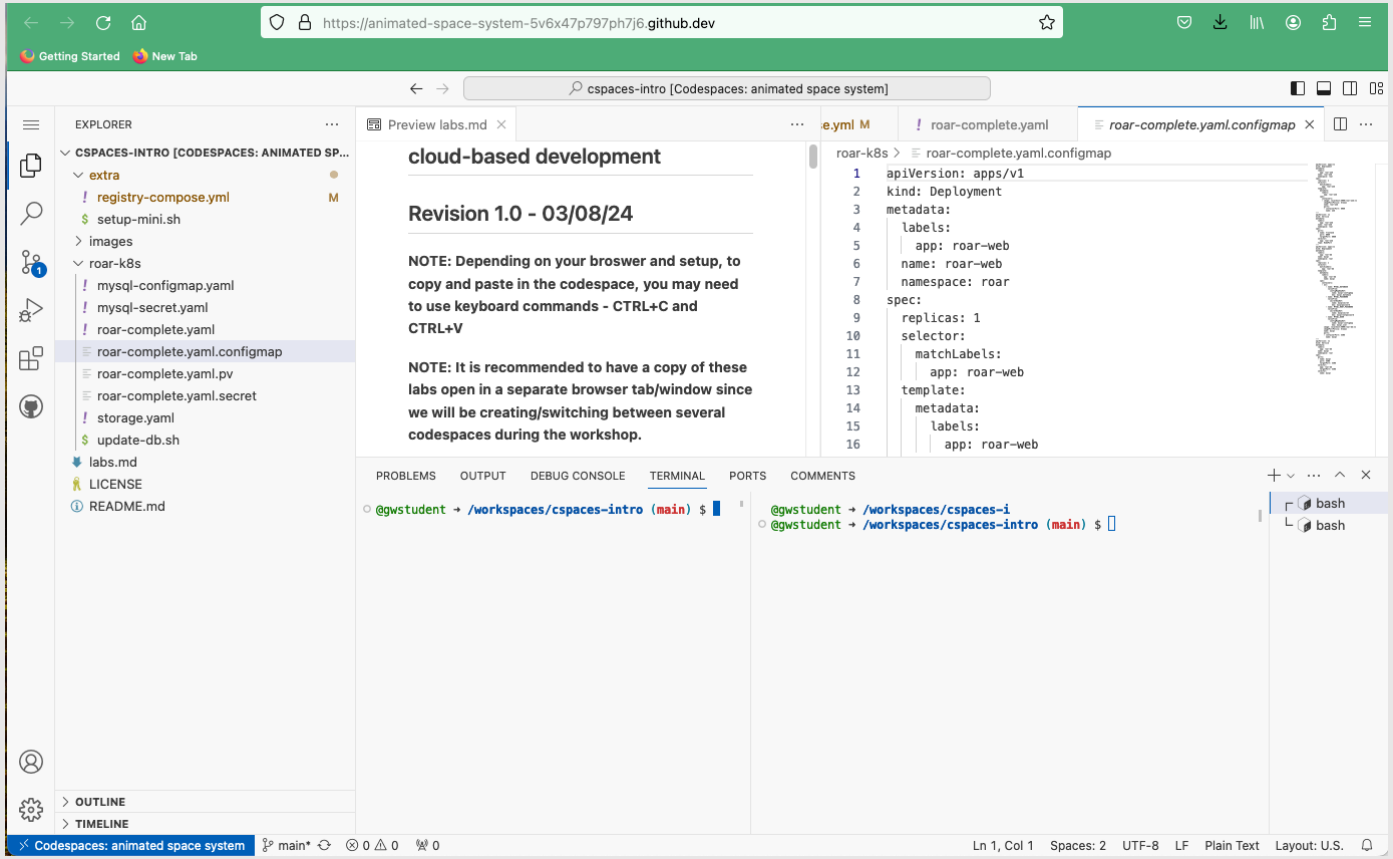
LIVE ONLINE TRAINING
Building a deployment pipeline with Jenkins 2
Manage continuous integration and continuous delivery to release software
Topic: System Administration


LIVE ONLINE TRAINING
Git Troubleshooting
How to solve practically any problem that comes your way
Topic: Software Development

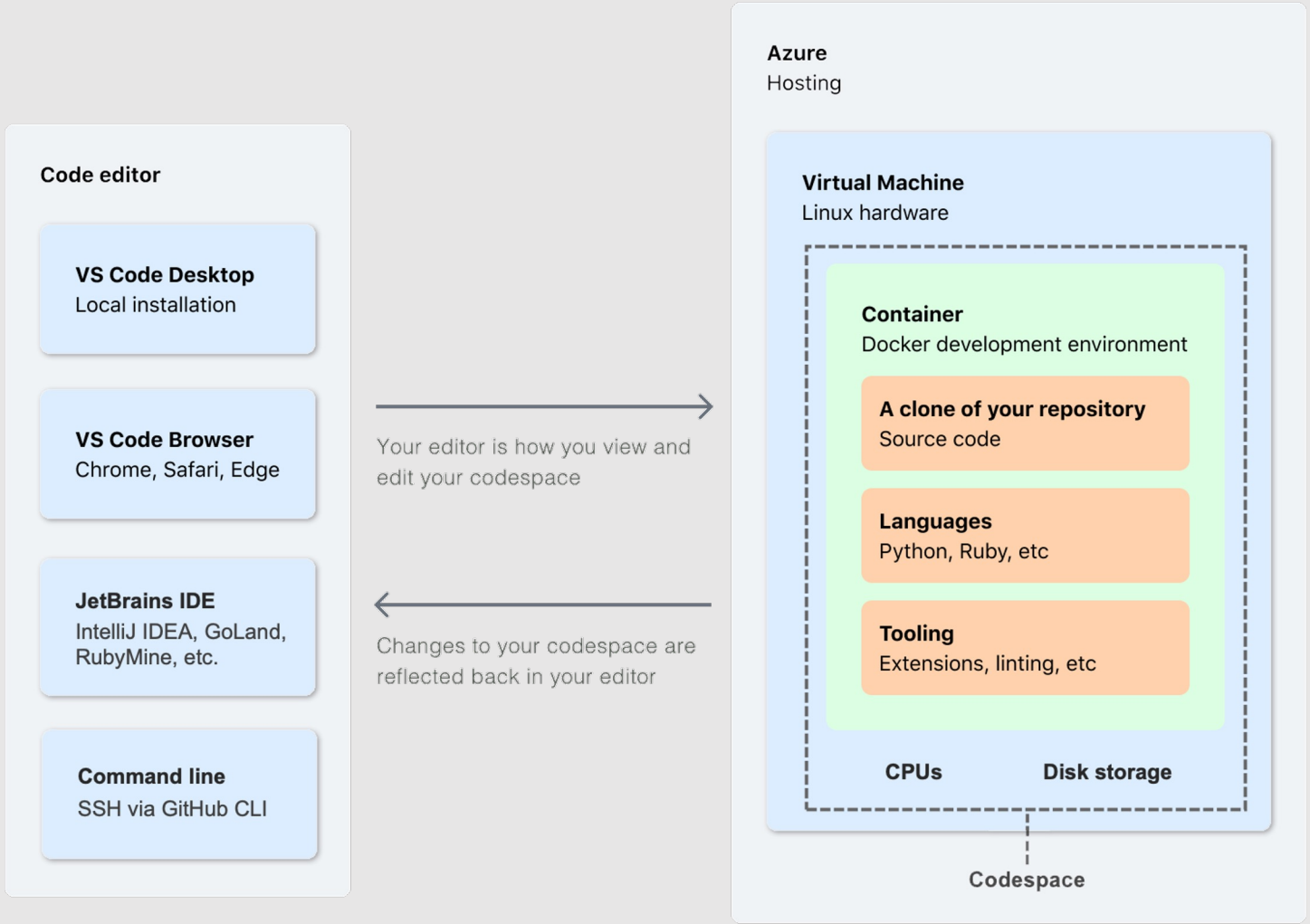

LIVE ONLINE TRAINING
Next Level Git - Master your workflow
Use Git to find problems, simplify working with multiple branches and repositories, and customize behavior with hooks
Topic: Software Development


What is a codespace?

- Virtual Machine running in the cloud
- Allows for reproducible, consistent development environments
  - Codespaces can be prebuilt
- Configure once and distribute easily
- Development environment can be a product
- Connects into favorite IDES or browser
- Editor running local - connects to codespace in cloud
- Can customize/install things and it all is only in codespaces - local system is not polluted
- 60 free hours/month/user



Codespace architecture



credit: https://docs.github.com/en/codespaces/overview

Codespace Environments

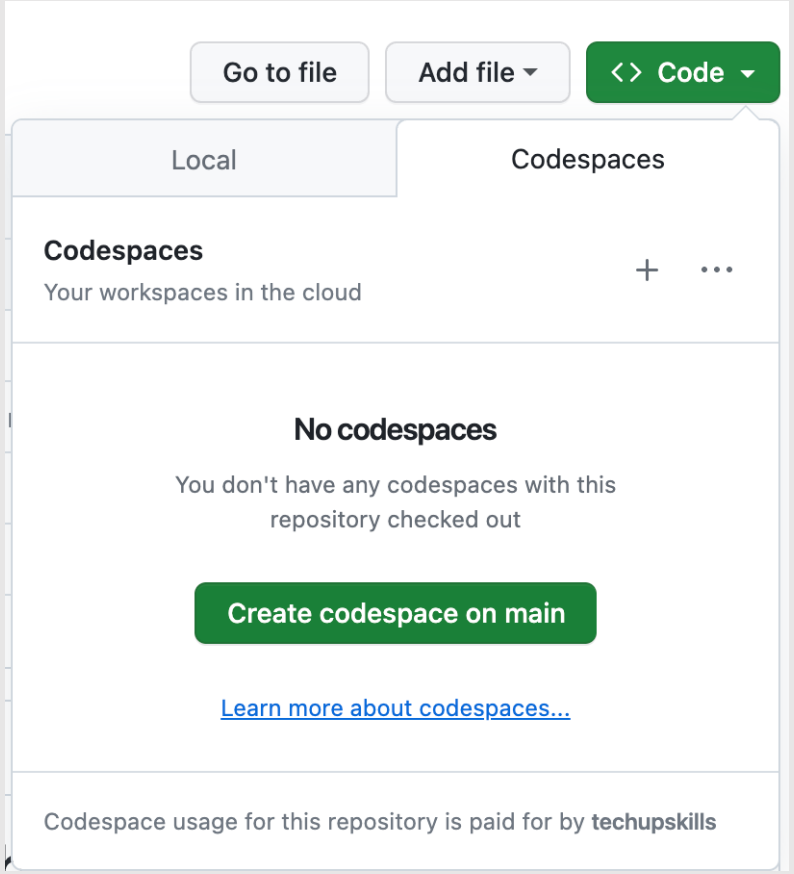
- Backend part of codespaces
- Where compute happens
  - compiling, debugging, restores, etc.
- Provides cloud-hosted environment
  - configured by codespaces
- Codespaces configures
  - source code
  - runtime
  - compiler
  - debugger
  - editor
  - custom dotfile configs
  - editor extensions
- Codespaces are geographically distributed so you can create a codespace near you

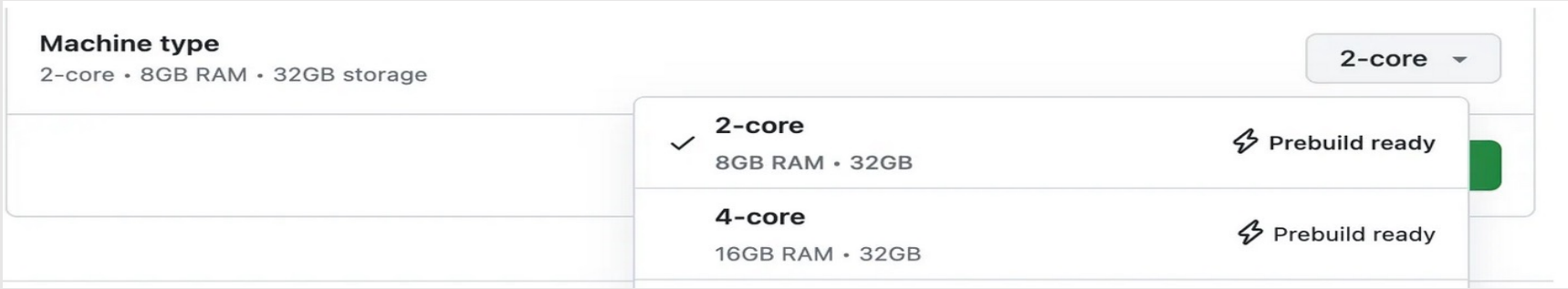
Supported IDEs

- Can develop in a codespace in any of the following:
  - Command shell via the GitHub CLI
  - A JetBrains IDE (via the JetBrains Gateway)
  - Visual Studio Code desktop app
  - Visual Studio Code in a browser

Codespace Lifecycle - Creation

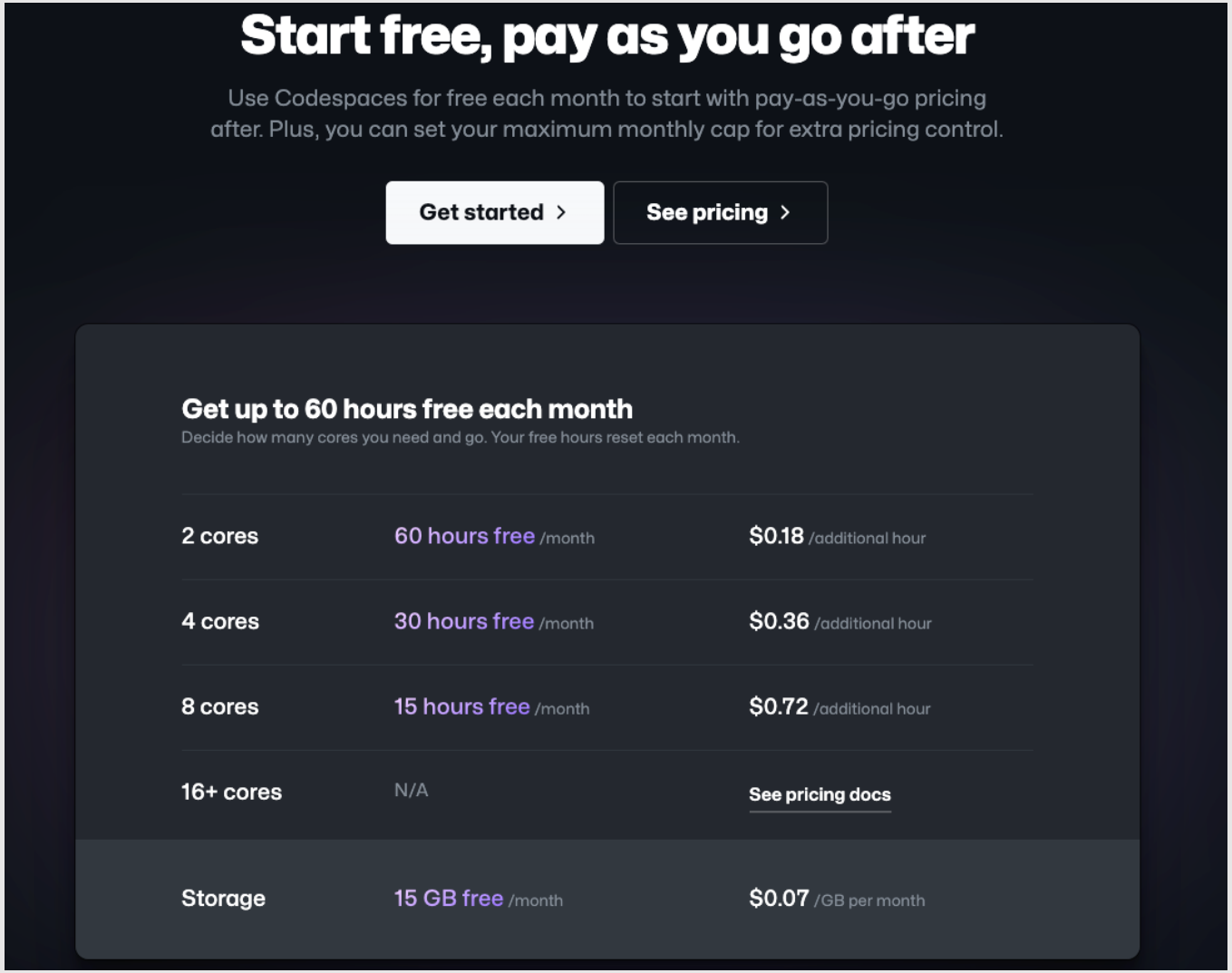
- Can choose to create new one or open existing one
- Like branches in Git
  - Can create a new codespace from a branch each time you develop something (fix, experment, etc.)
    - » Push changes regularly
  - Can have long-lived one for developing a feature
    - » Pull changes regularly



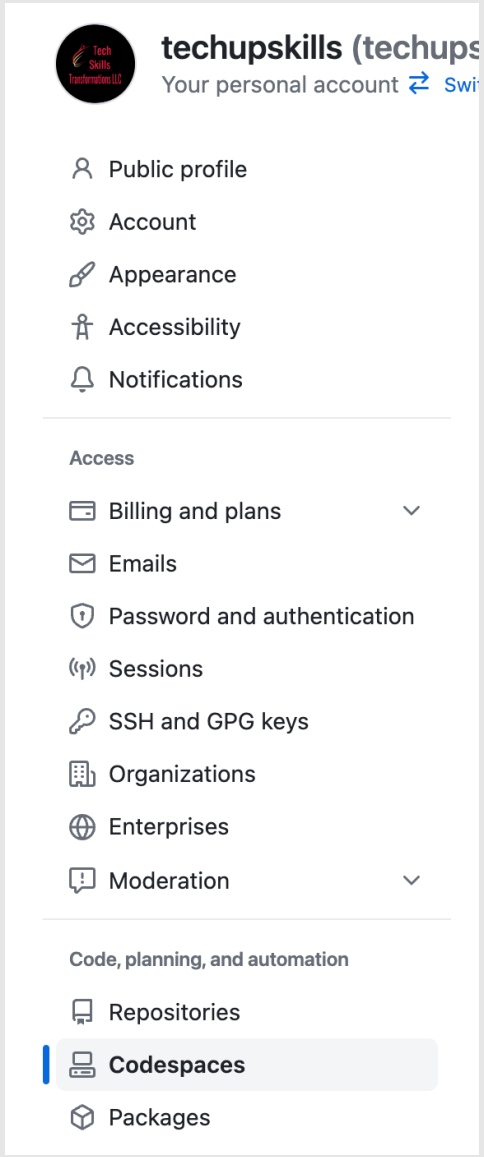


Codespace Lifecycle - Active

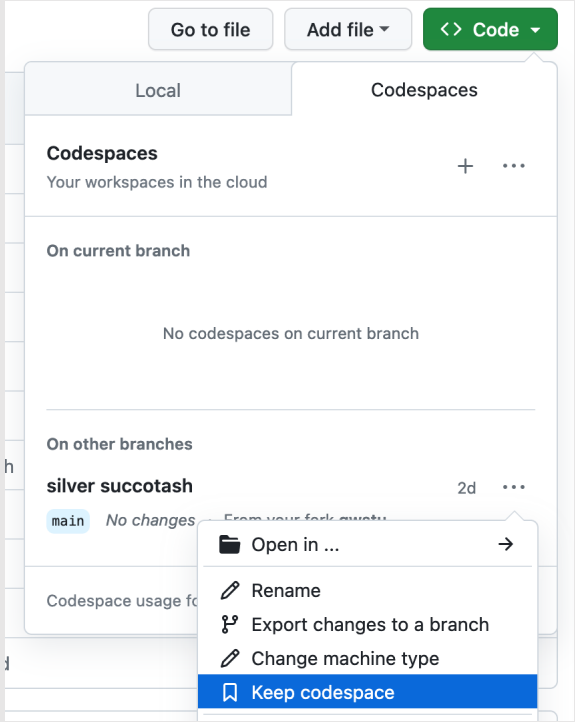
- Begins when you create a codespace
- Ends when you delete it
- Can disconnect and reconnect to an active one w/o affecting running processes
- Can stop and restart w/o losing changes made to project
- Limits for how many you can run at once
- Limits for how many you can create
- Limits vary based on factors like amount of use



Codespace Lifecycle - Deletion



- Codespaces that are inactive for default 30 days will be deleted
- Inactivity period is configurable (1 - 30 days)
  - 0 = delete immediately on stop
- Can be set at github.com/settings/codespaces
- Email before deletion
- Option to keep codespace indefinitely in main menu
  - Retained until manually deleted
  - May not be available if organization policy set



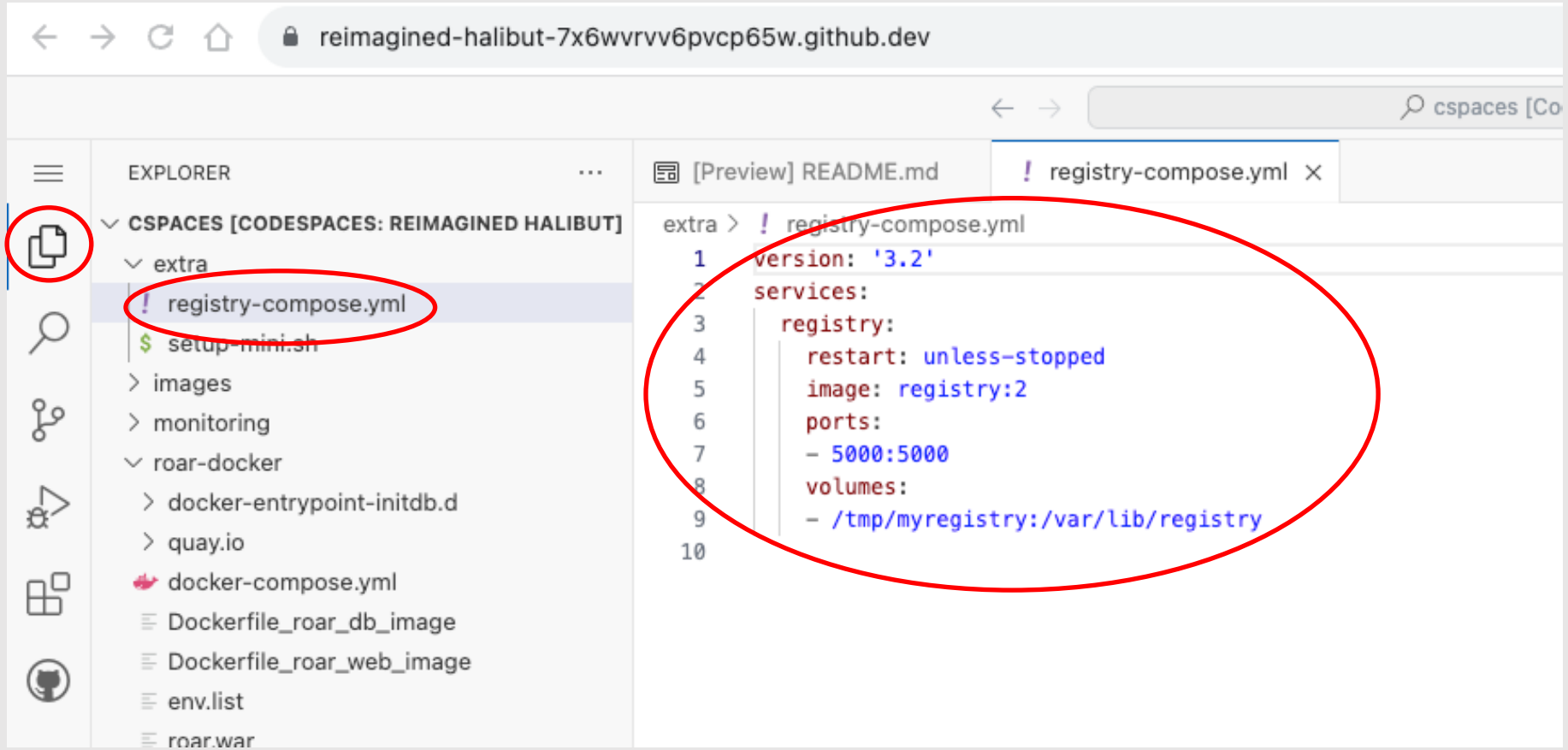
Default retention period

Inactive codespaces are automatically deleted 30 days after the last time they were stopped. A shorter retention period can be set, and will apply to all codespaces created going forward. The default and maximum value is **30** days. Learn about retention setting

30 days Save

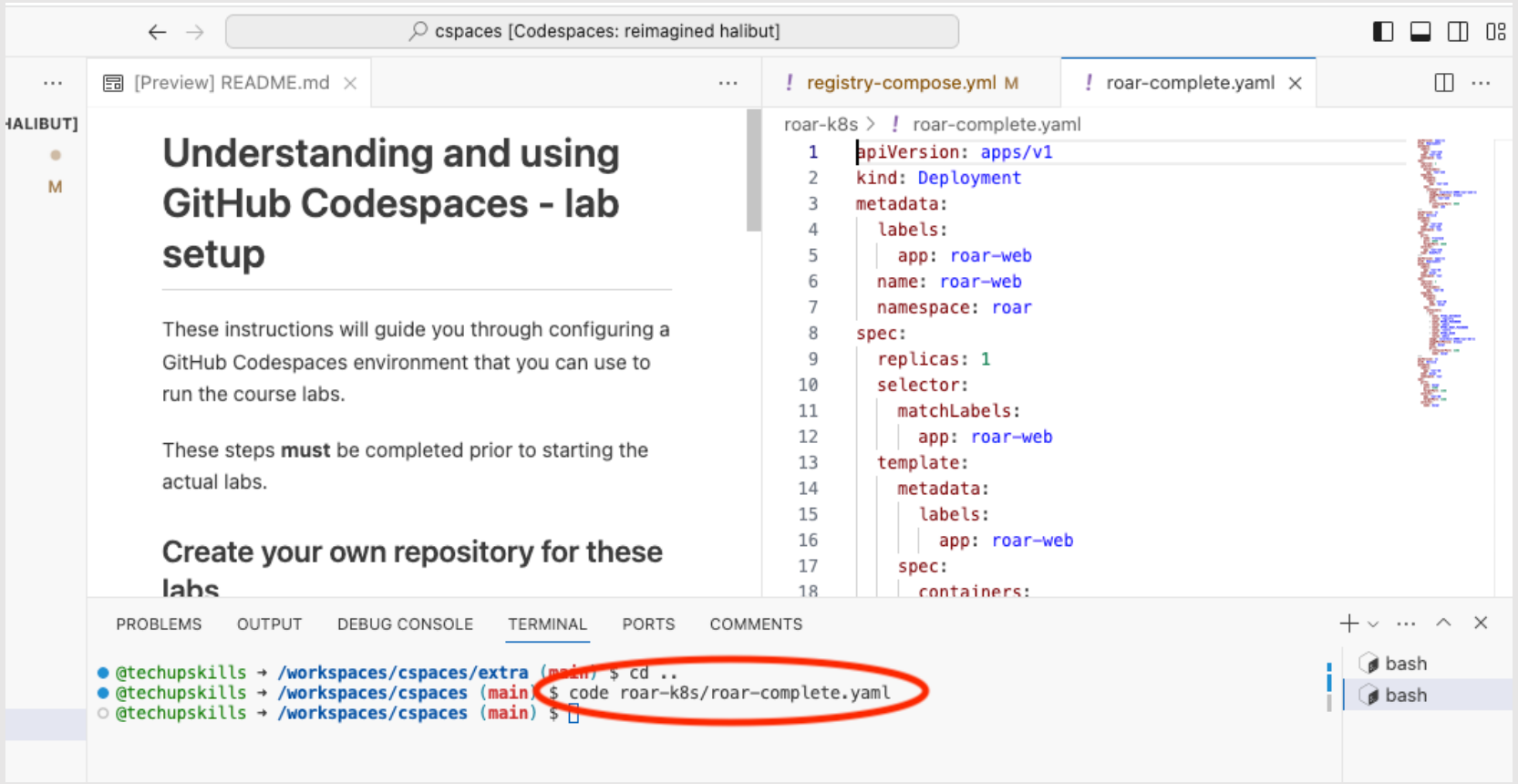
Working with files in the editor

- Click on Explorer icon in activity bar
- Select file
- Edit
- Automatically saved



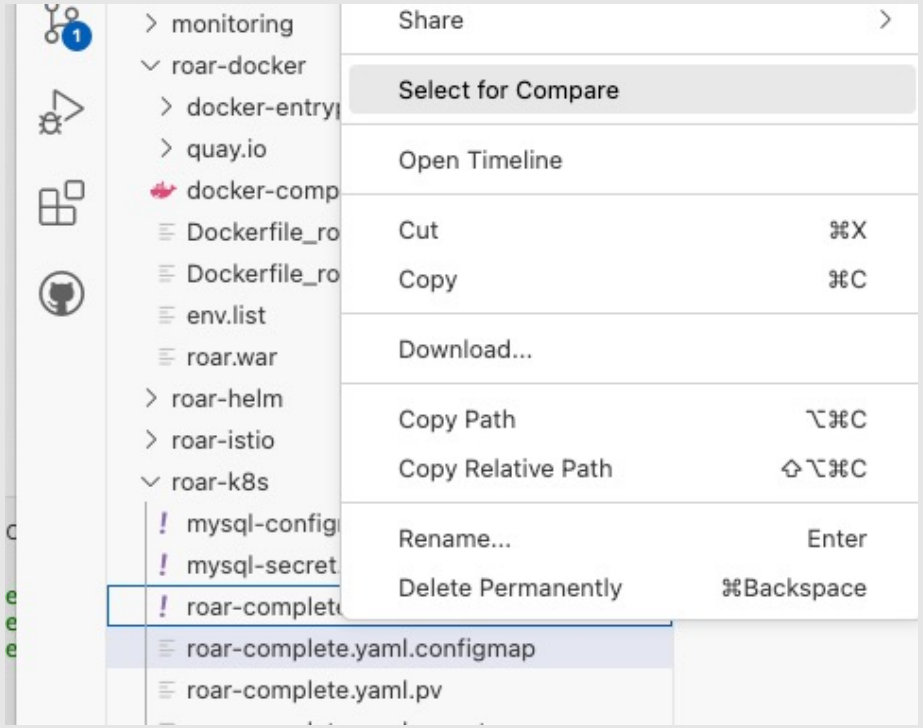
Opening files from terminal

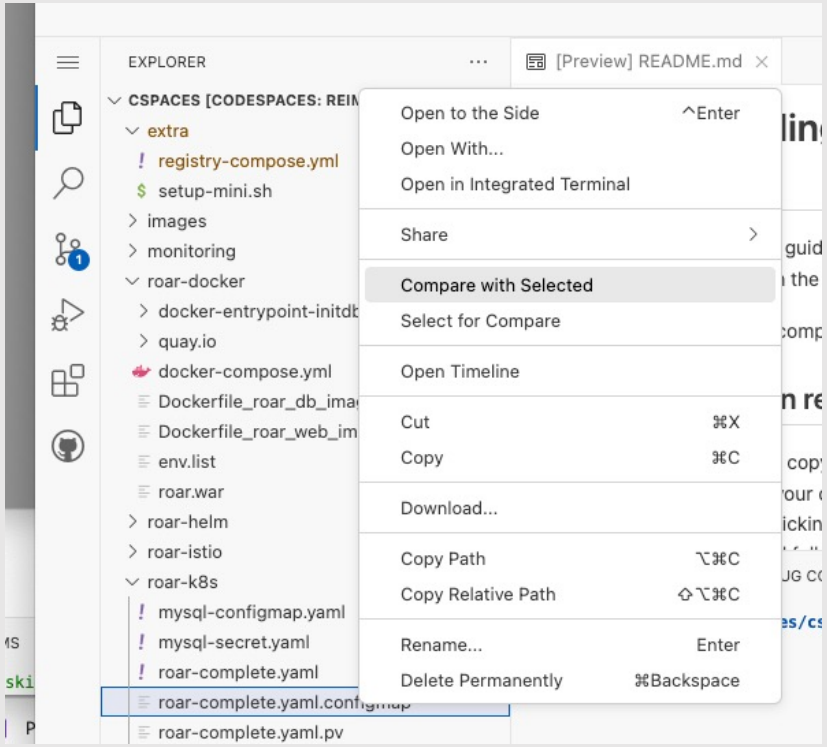
- "code" command invokes built-in editor
- paths are relative to /workspaces/<repo>



Selecting files to compare in editor

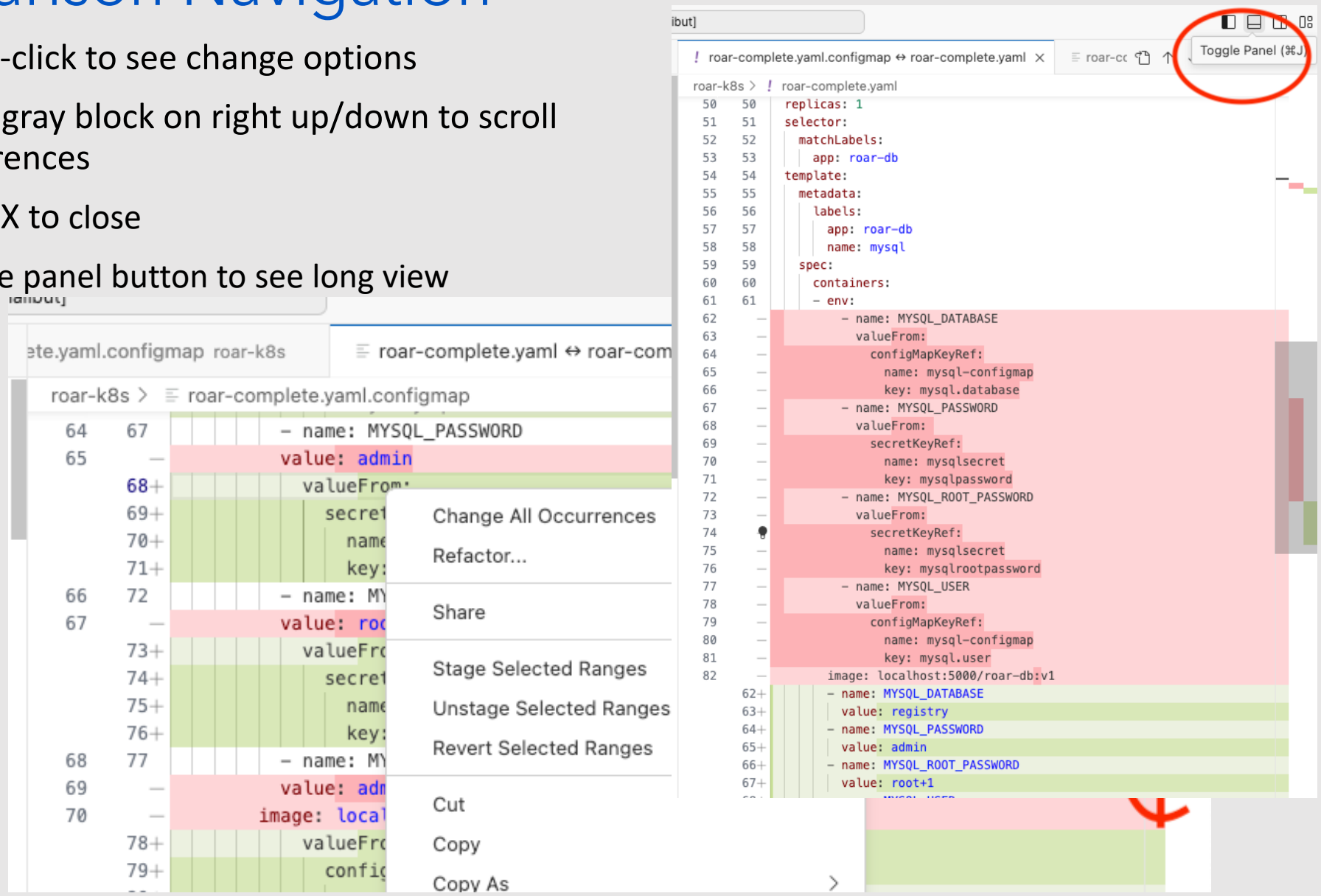
- Select first one "for Compare"
- Select second and right-click "Compare with Selected"
- Also can use "code -d <file1> <file2> in terminal





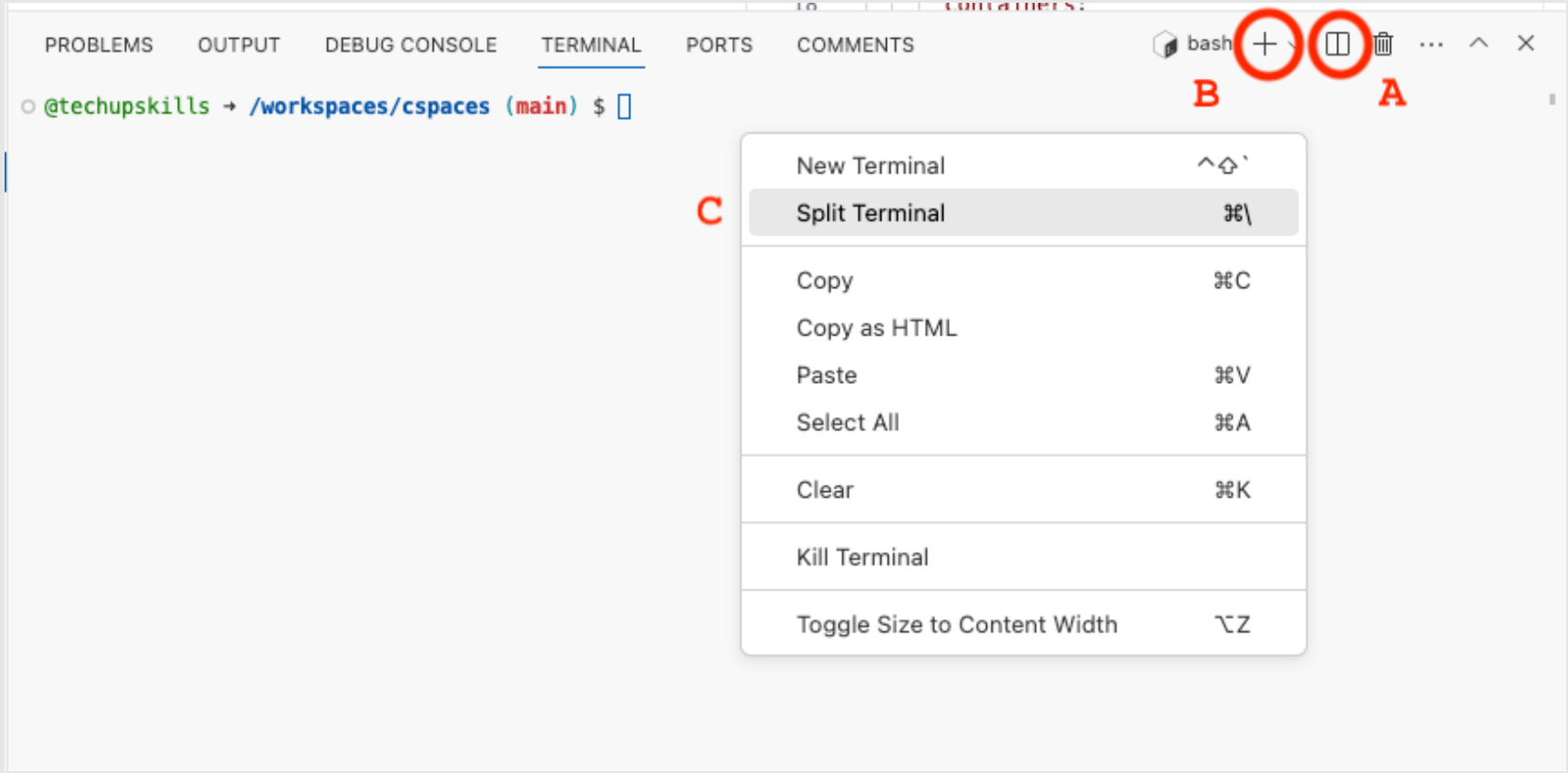
Comparison Navigation

- Right-click to see change options
- Drag gray block on right up/down to scroll differences
- Click X to close
- Toggle panel button to see long view



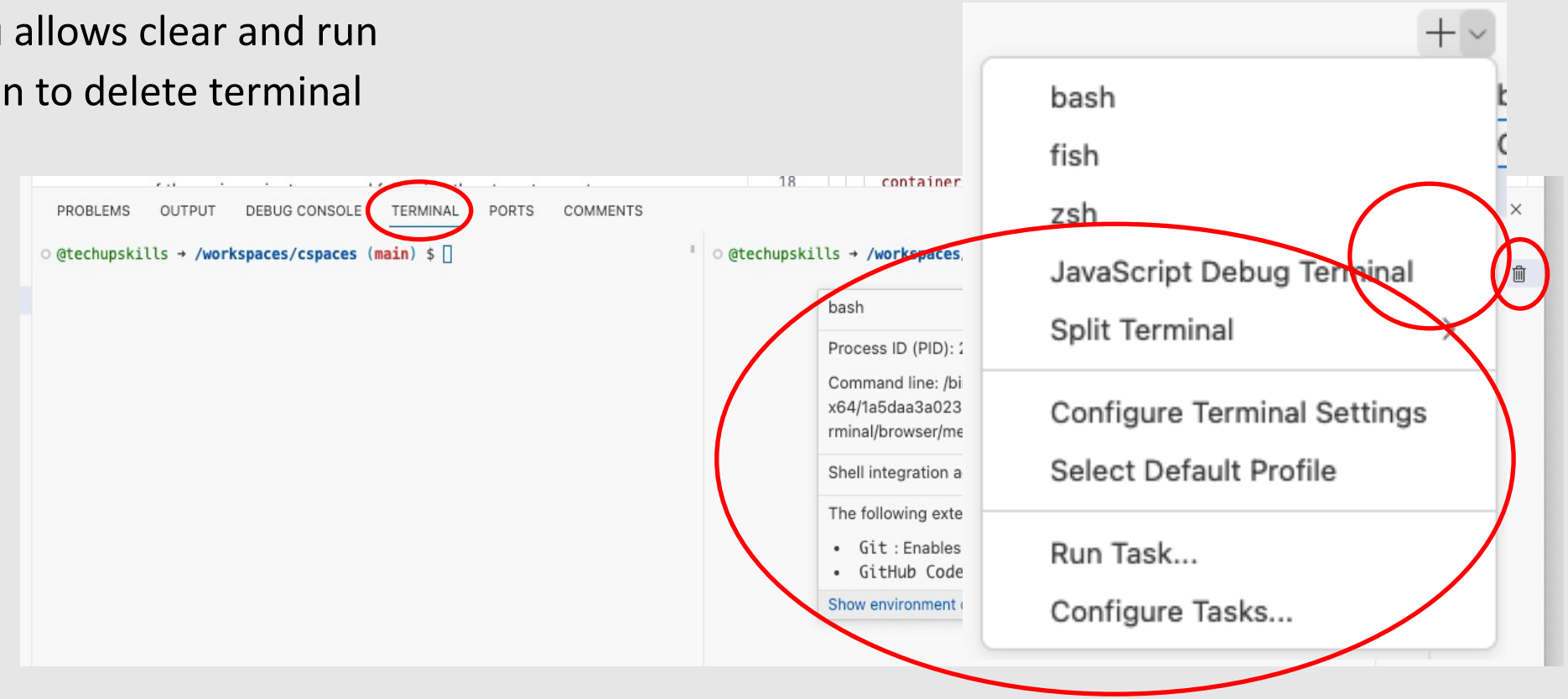
Adding a second terminal

- Can click split icon - A
- Can add a new terminal - B
- Can right-click and select "Split Terminal" - C



Managing terminals

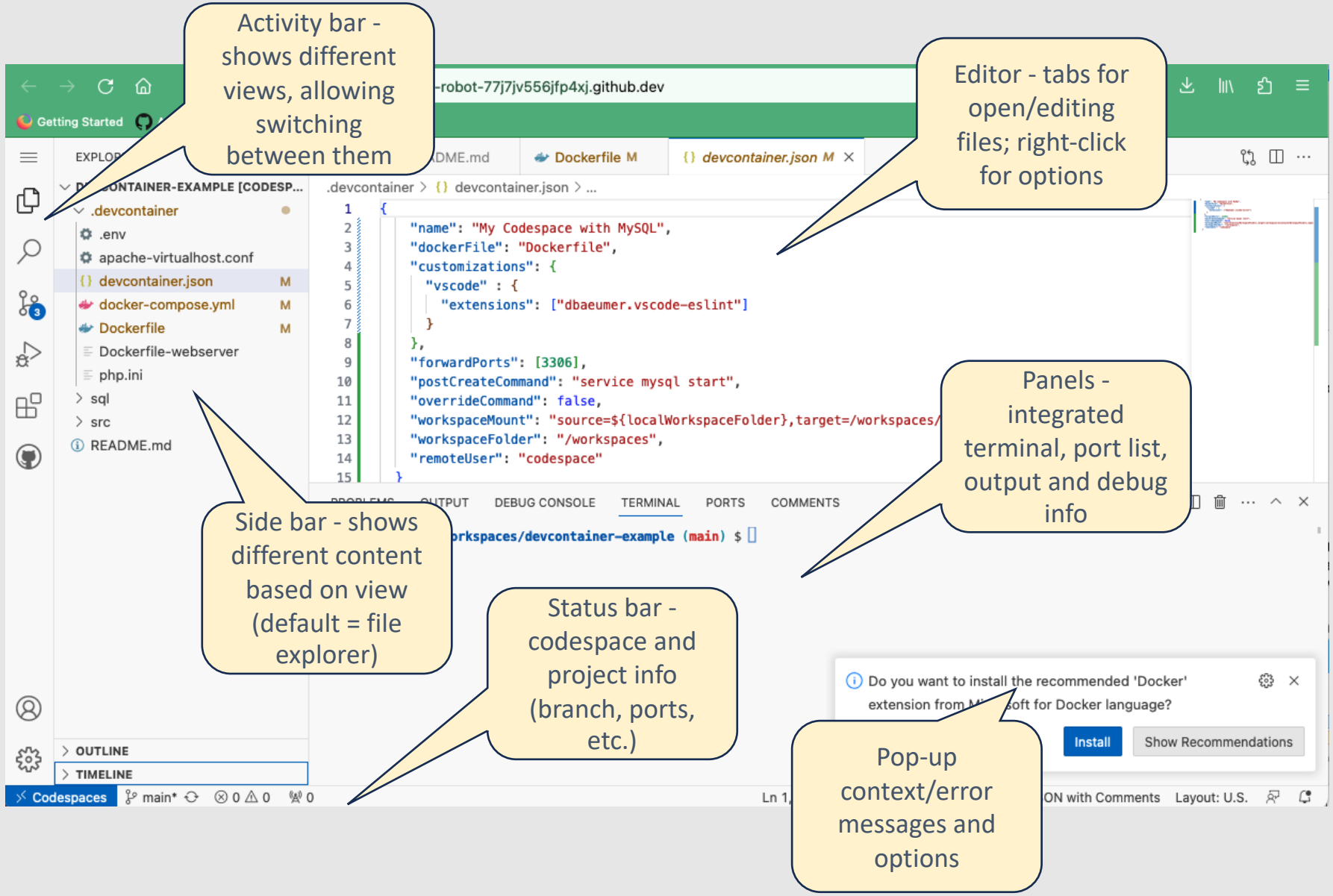
- Select "TERMINAL" tab
- "List" of terminals on right side
- Hover over entry to see details
- + to add new terminal
- ... menu allows clear and run
- Trash can to delete terminal



Lab 1 – Exploring Codespaces

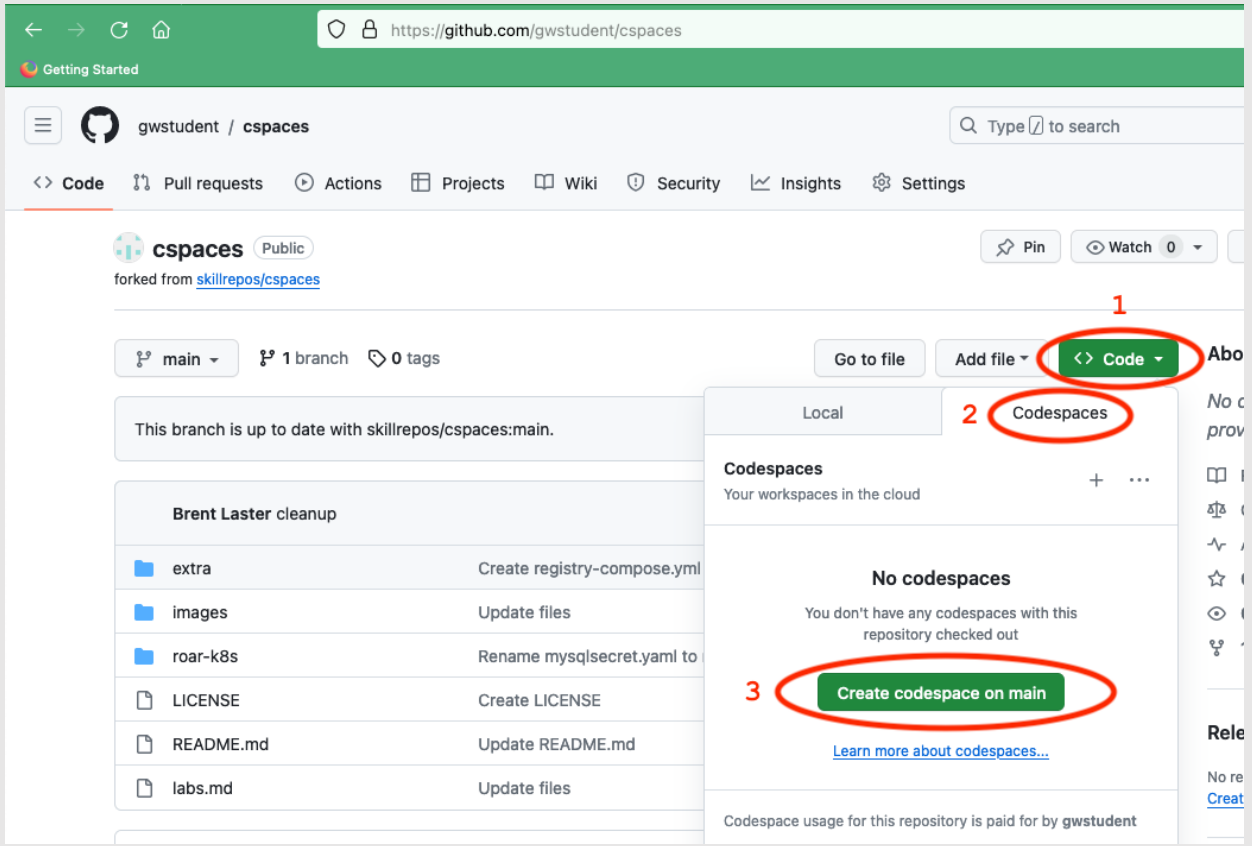
Purpose: In this lab, we'll start to learn our way around a codespace.

Codespace in a browser - components

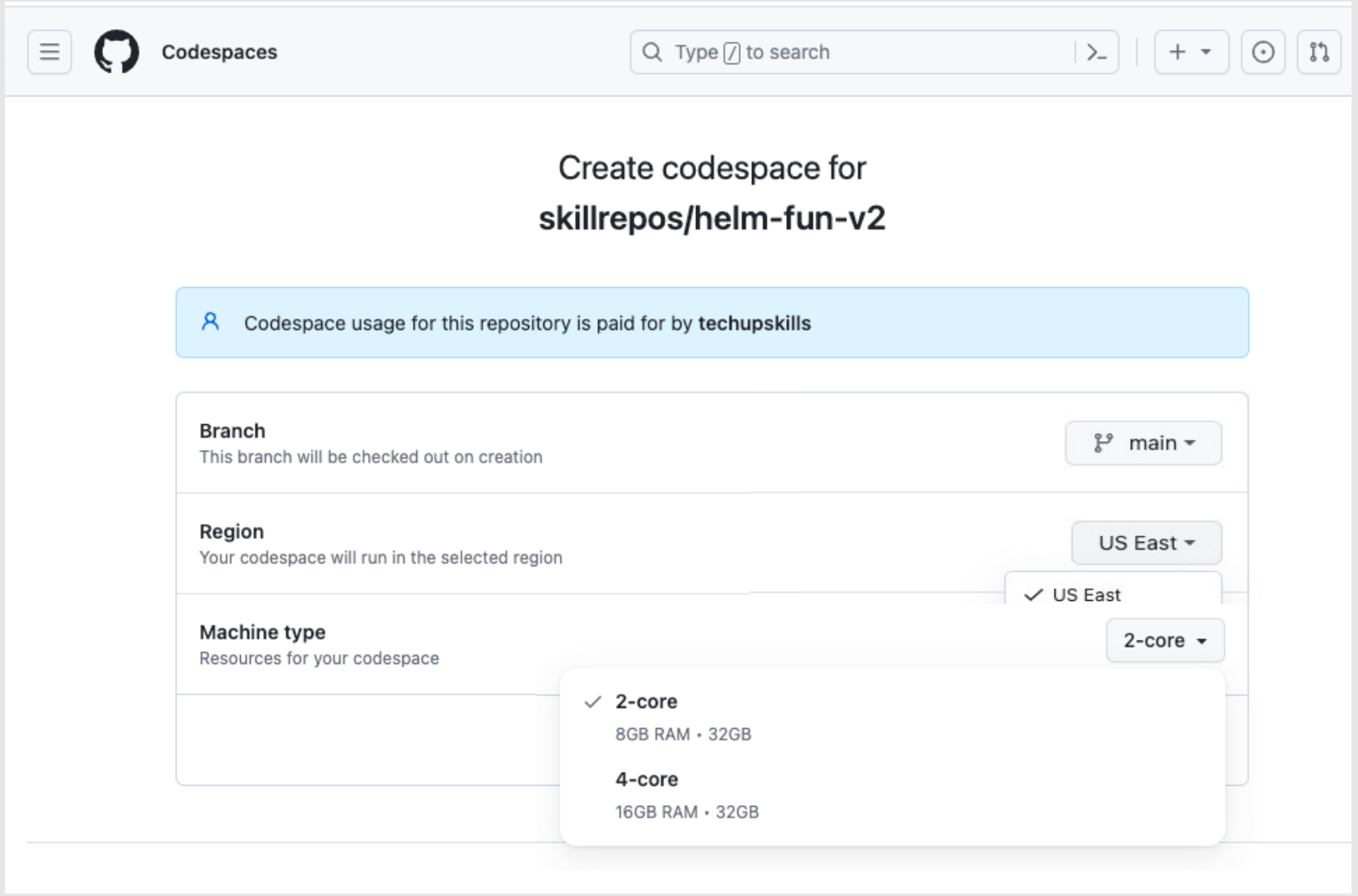


Create a new codespace

- Click "<> Code" button
- Click on "Codespaces" tab
- Select "Create codespace on main"

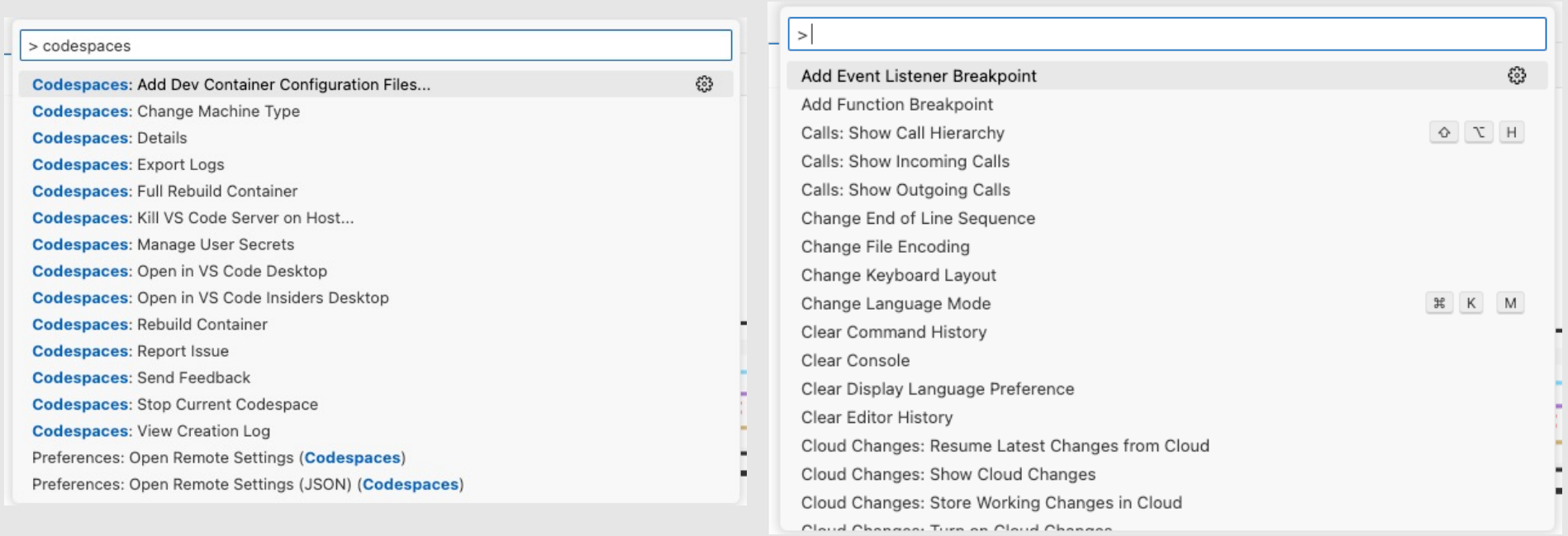


Options for configuring Codespaces

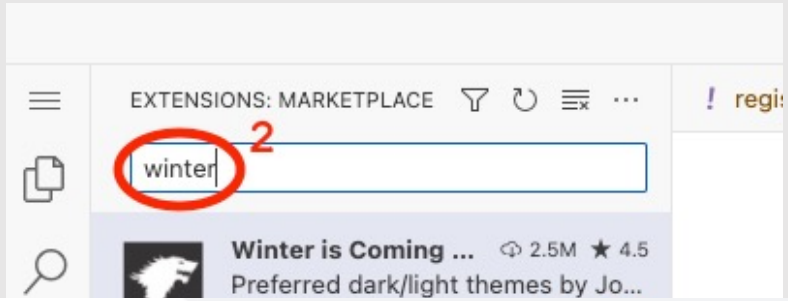


Codespaces Command Palette

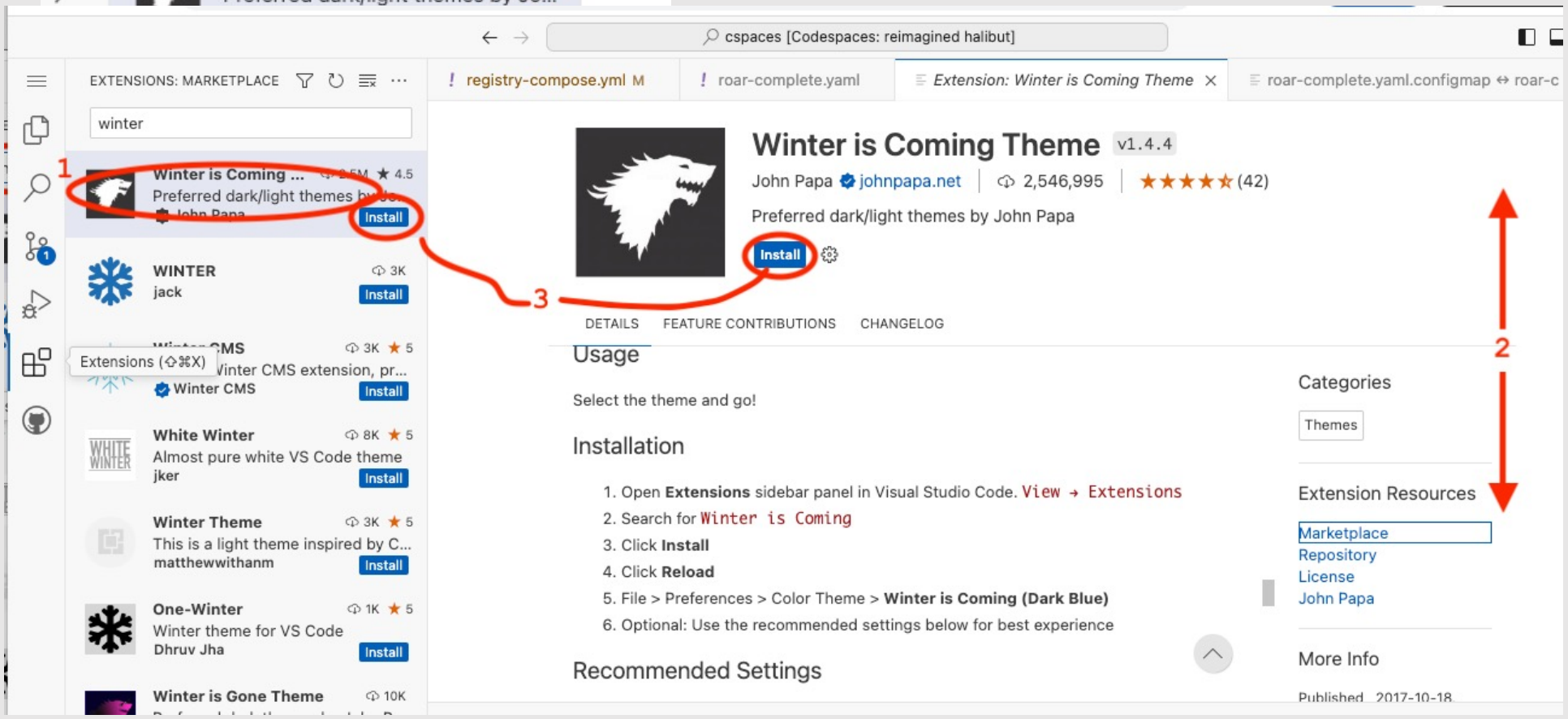
- Can use the Command Palette of Visual Studio Code
- Allows you to access many commands for Codespaces *and* VS Code
- Execute editor commands, open files, search, etc.
- Typically Shift+Cmd/Ctrl+P, but that shortcut is already taken in Firefox
- Can use F1 instead



Finding and installing extensions

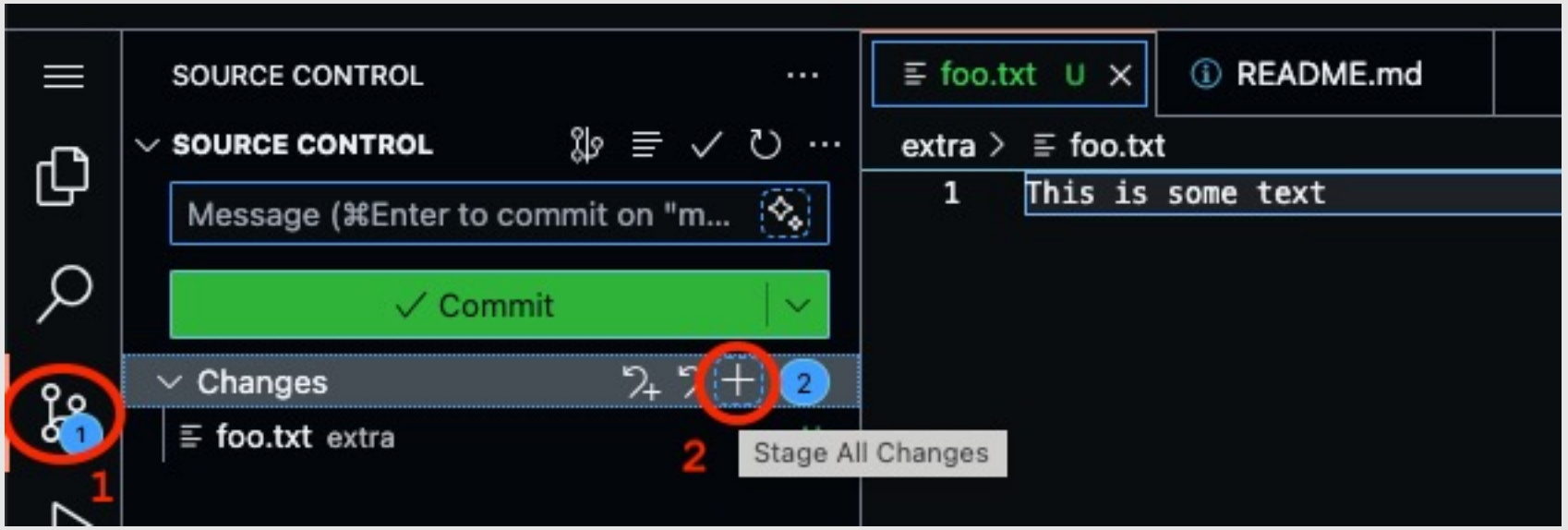


- Select Extensions icon in activity bar
- Search for extension
- Install
- Some may add icons in activity bar



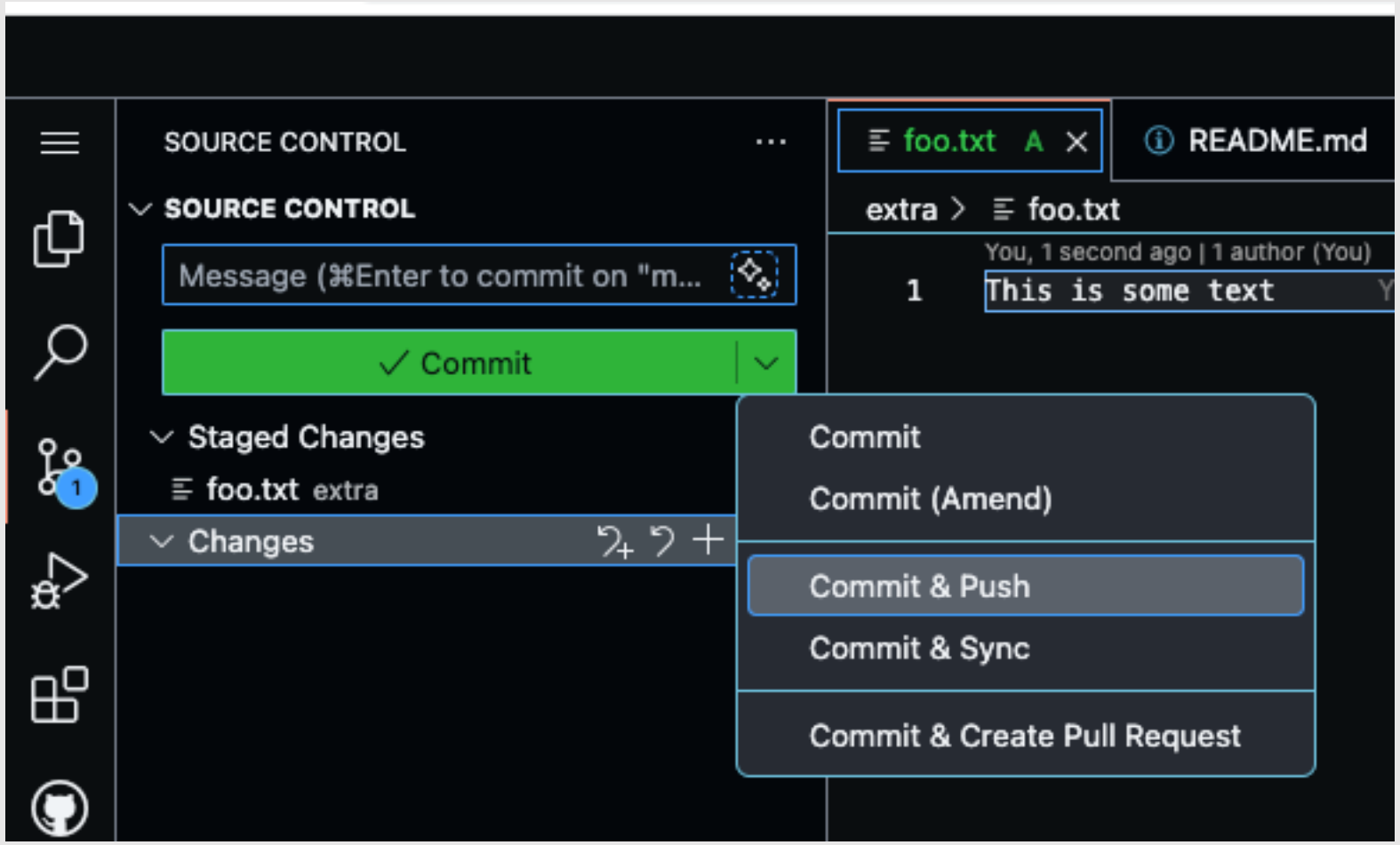
Source control - staging changes

- Make changes
- Click on source control icon
- Use "+" to explicitly stage changes
  - Can also wait and be prompted on commit



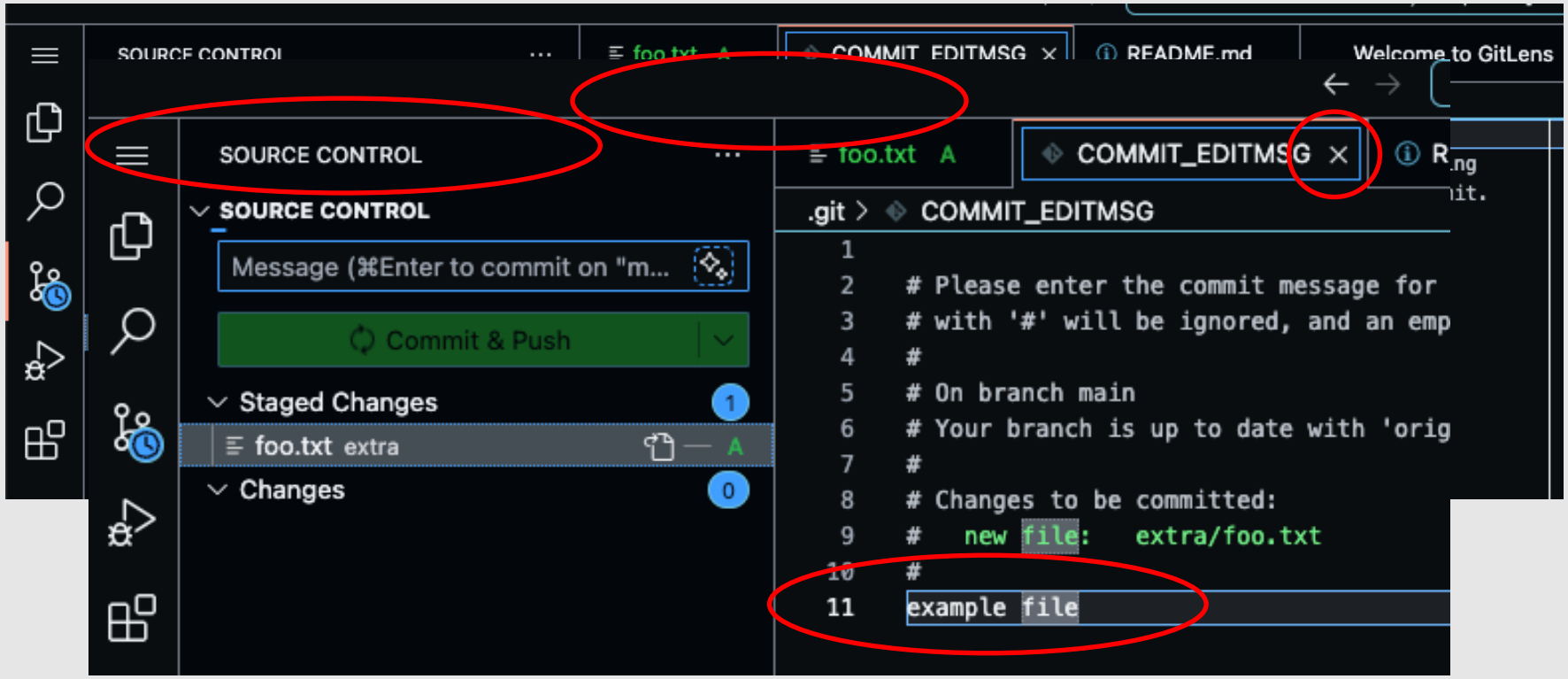
Source control - committing changes

- After staging, Commit or Commit & Push, etc.



Source control - completing commit

- Supply commit message in "Message" box, or
- Git editor will pop up for commit message
- Complete and close via "x"

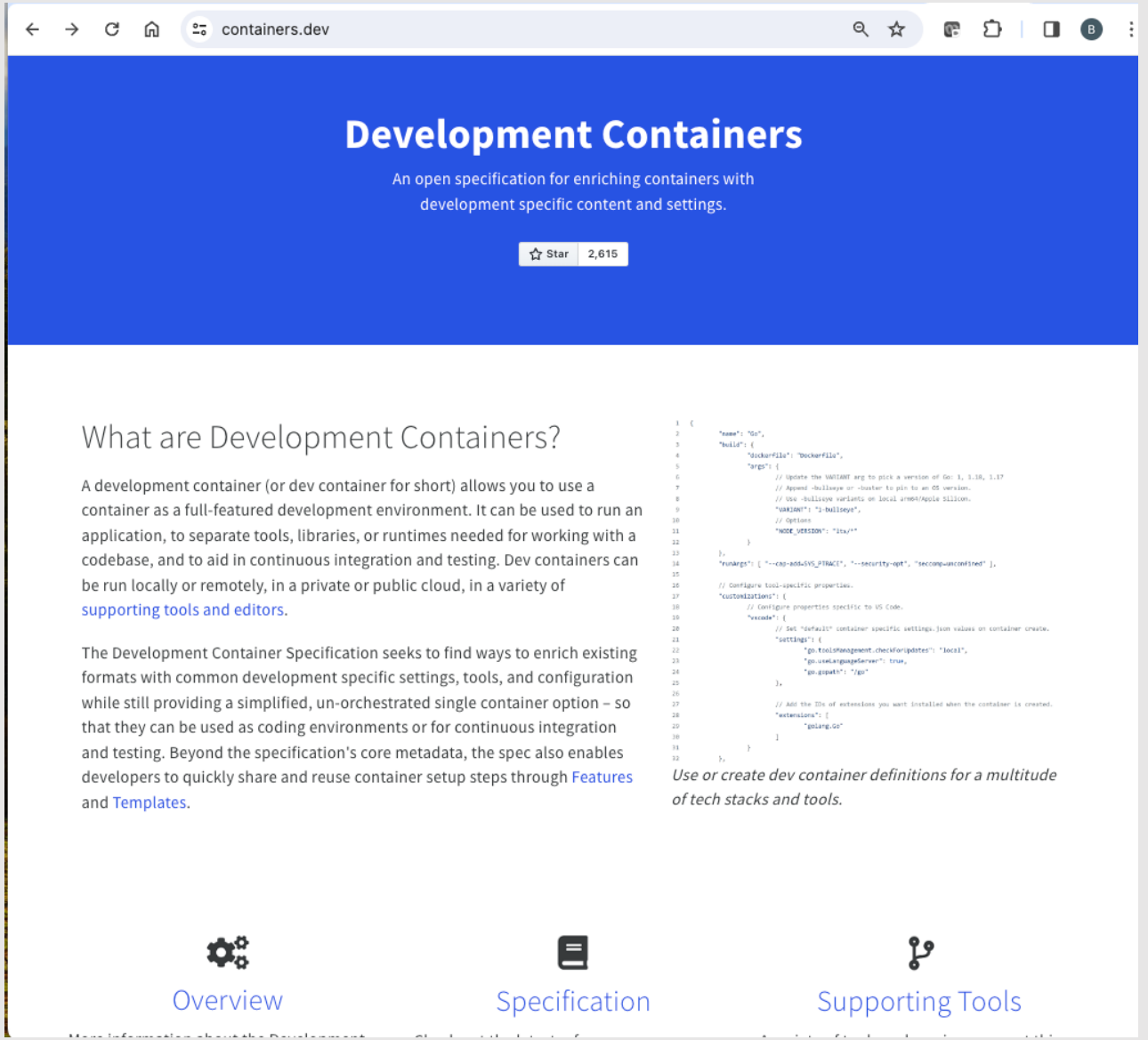


Lab 2 – Customizing Codespaces

Purpose: In this lab, we'll see how to customize a codespace.

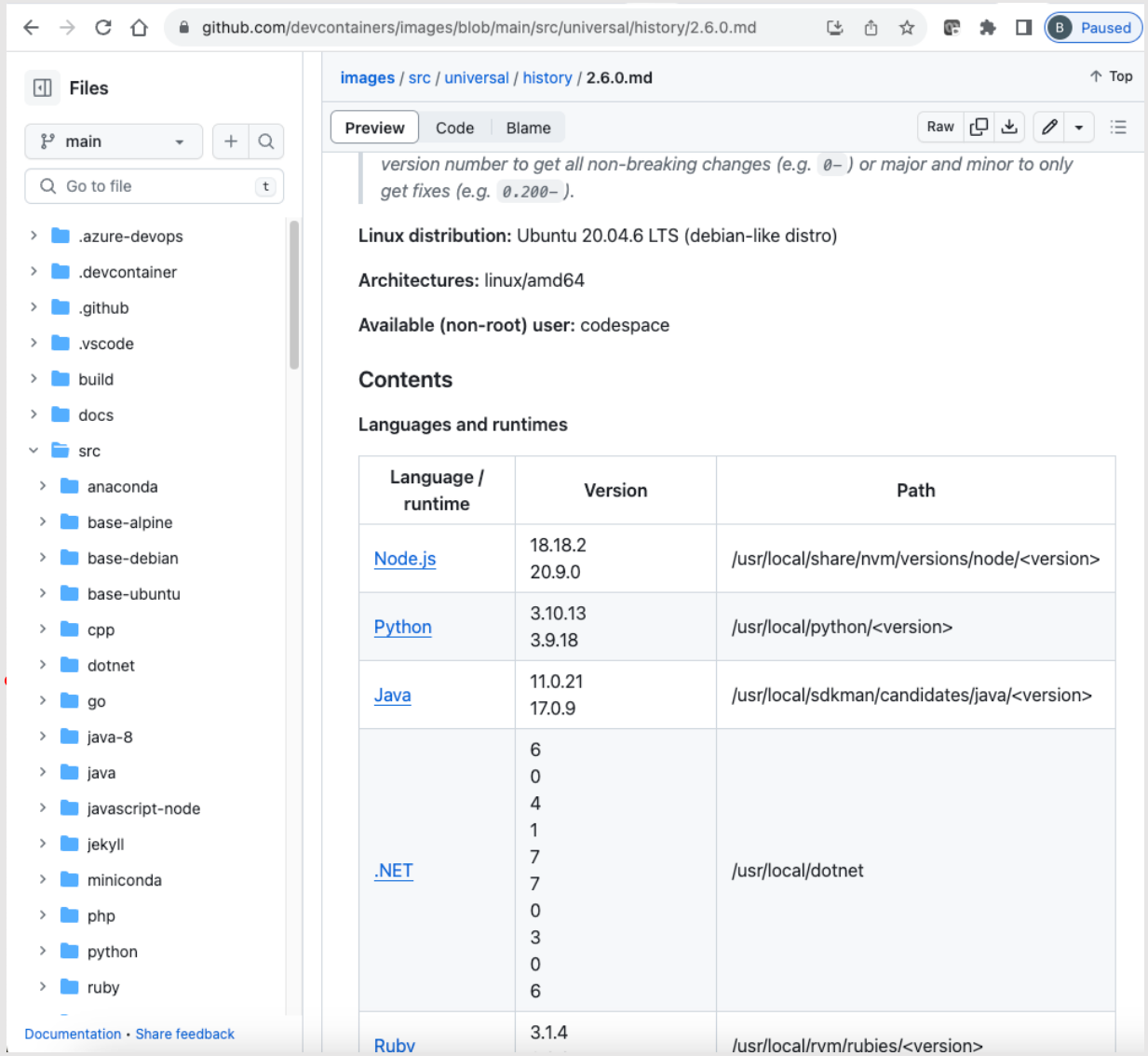
Development Containers for VS Code & Codespaces

- Development Containers - Visual Studio Code feature
- Spec at https://containers.dev
- Allows you to use Docker to setup your dev environment
- Environment in a codespace is created using a development container (dev container) hosted on a VM
  - Config for dev container in repo = config for env
  - Do your development inside the container
  - Install all of the tools you need into container
  - Use that for your dev environment



Default dev container config

- Used if you don't add a dev container config to repo
- or if config does not specify a base image
- Based on Linux image with runtimes for latest language releases of
  - Python
  - Node
  - PHP
  - Java
  - Go
  - C++
  - Ruby
  - .NET Core/C#
- Tools
  - Git
  - GitHub CLI
  - yarn
  - openssh
  - vim
- Interfaces
  - JupyterLab
  - Conda

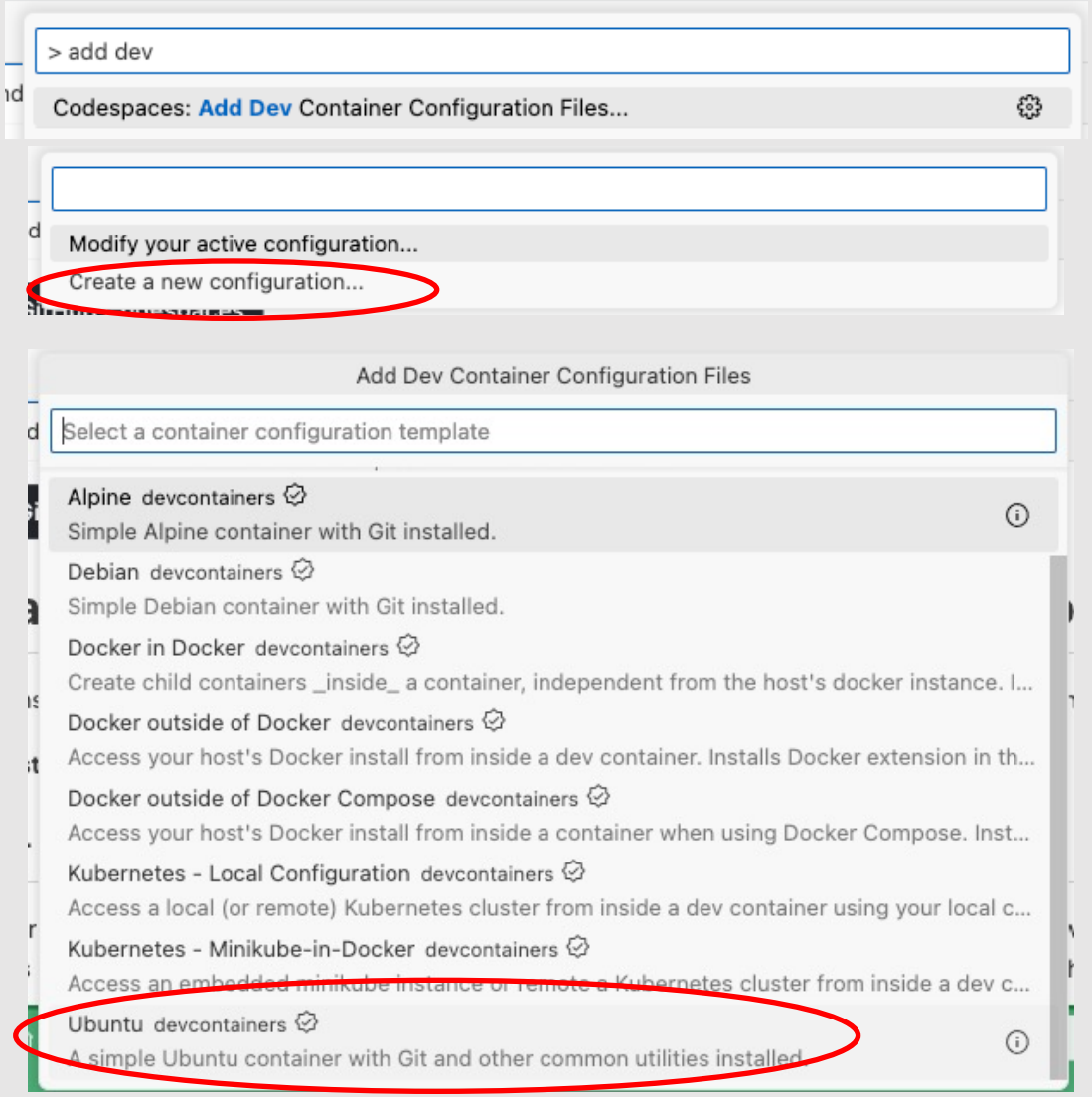


Codespace Customizations

- Fully customizable on per project level
- Done by including devcontainer.json file in repo
- Like VS Code Dev Containers dev
- Example customizations
  - Setting Linux-based OS
  - Auto-install of tools, runtimes, and frameworks
  - Forwarding ports
  - Setting env vars
  - Configuring editor settings & installing extensions

Using a predefined dev container config

- Can create dev container config from list of predefined configs
- Access Command Palette
- "add dev"
- Select Codespaces: **Add Dev Container Configuration Files**



Components of a dev container - devcontainer.json

- File to describe how VS Code should start/attach to the container and what to do after connecting
- Can use an image as a starting point

.devcontainer > {} devcontainer.json > ...

```
3   {
4       "name": "Ubuntu",
5       "image": "mcr.microsoft.com/devcontainers/base:jammy",
```

- Can also
  - install additional tools
  - install extensions
  - forward/publish ports
  - run commands
  - and more...
- Can exist as a standalone file in root of project or in .devcontainer folder if needs other files to build (Dockerfile, docker-compose.yml)

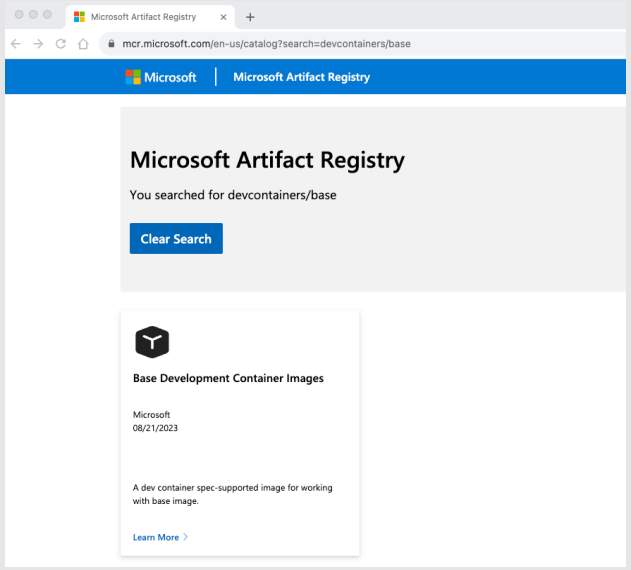
Using the devcontainer.json file

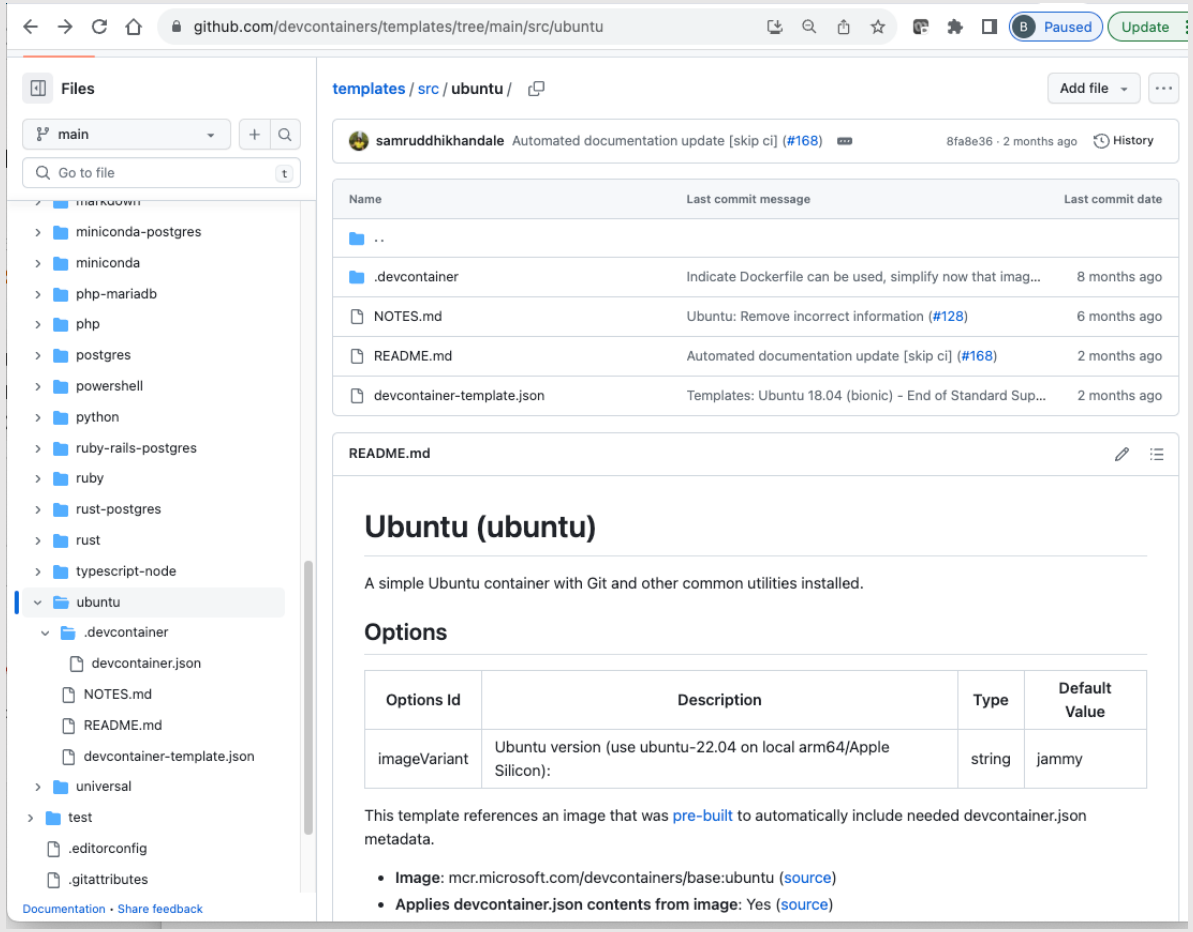
- Use the file to provide *customization* not *personalization*
- Include items in devcontainer.json file that anyone using codespace would need - i.e. linters
- Think standardization
- Avoid adding user interface themes or decorators (i.e. personal choices)
- Can personalize using dotfiles and Settings Sync

Items in devcontainer.json - images

- Can use an image as a starting point for devcontainer.json
- Image artifacts at mcr.microsoft.com/devcontainers
- Image source/templates at github.com/devcontainers/templates
- Also, community ones at github.com/devcontainers-contrib/templates

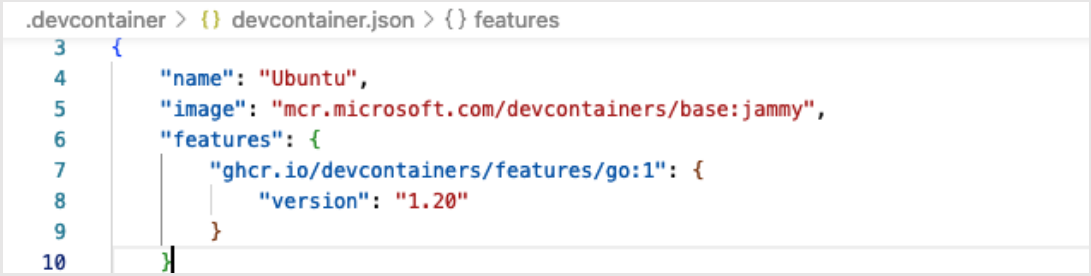
```
.devcontainer > {} devcontainer.json > ...
3   {
4       "name": "Ubuntu",
5       "image": "mcr.microsoft.com/devcontainers/base:jammy",
```

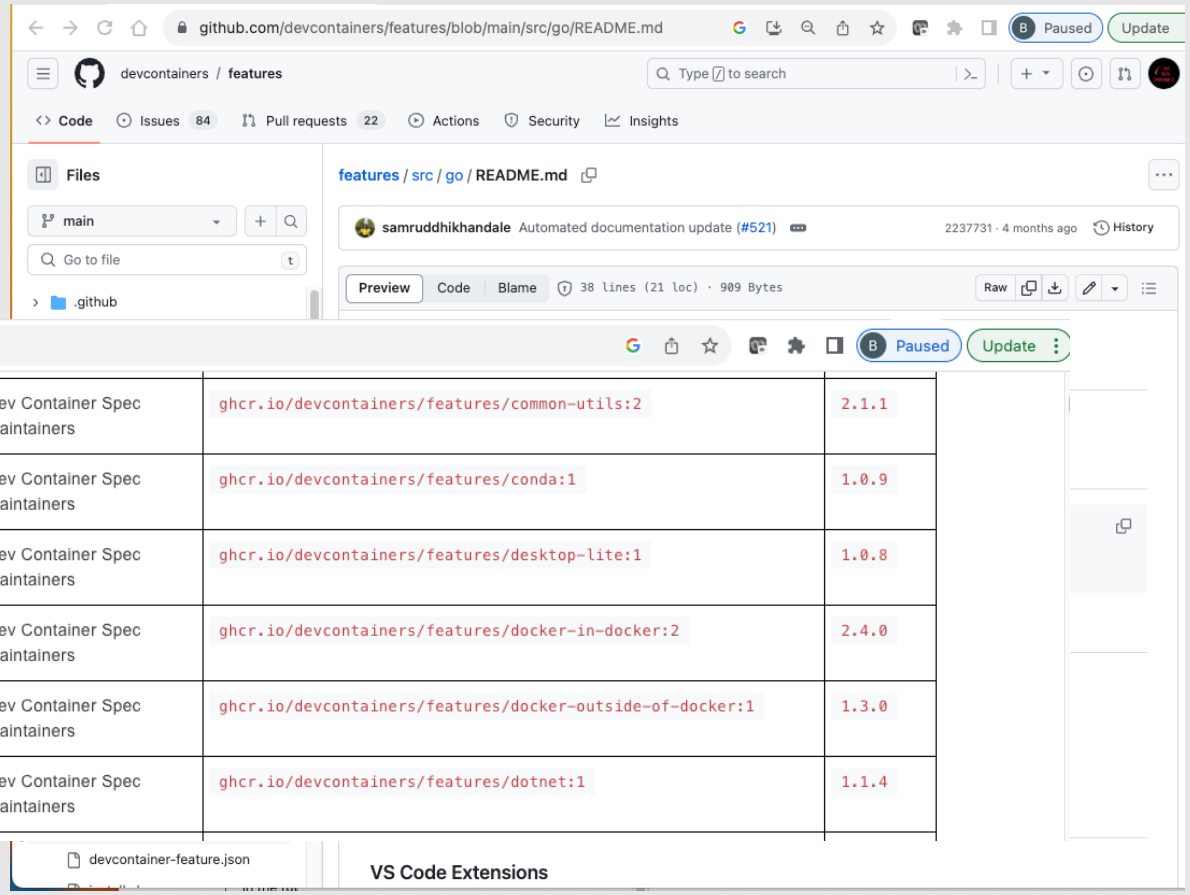




Items in devcontainer.json - features

- Way to easily add language/tool/CLI to a dev container
- Contain install scripts
- Feature source/templates at github.com/devcontainers/features
- Also community ones at github.com/devcontainers-contrib/features
- See also containers.dev/features

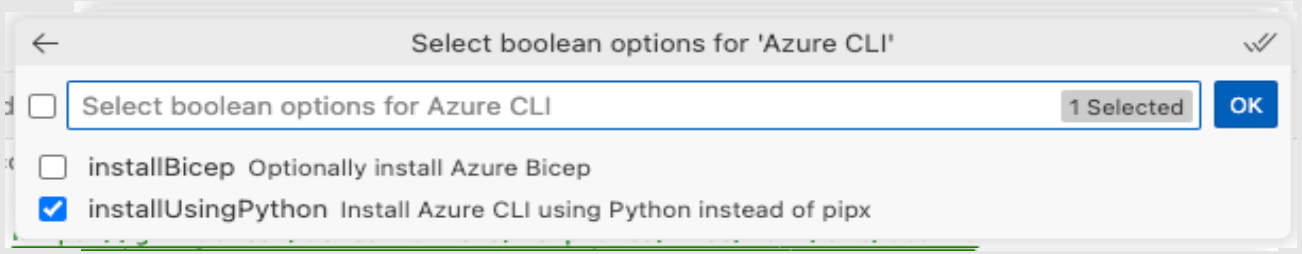


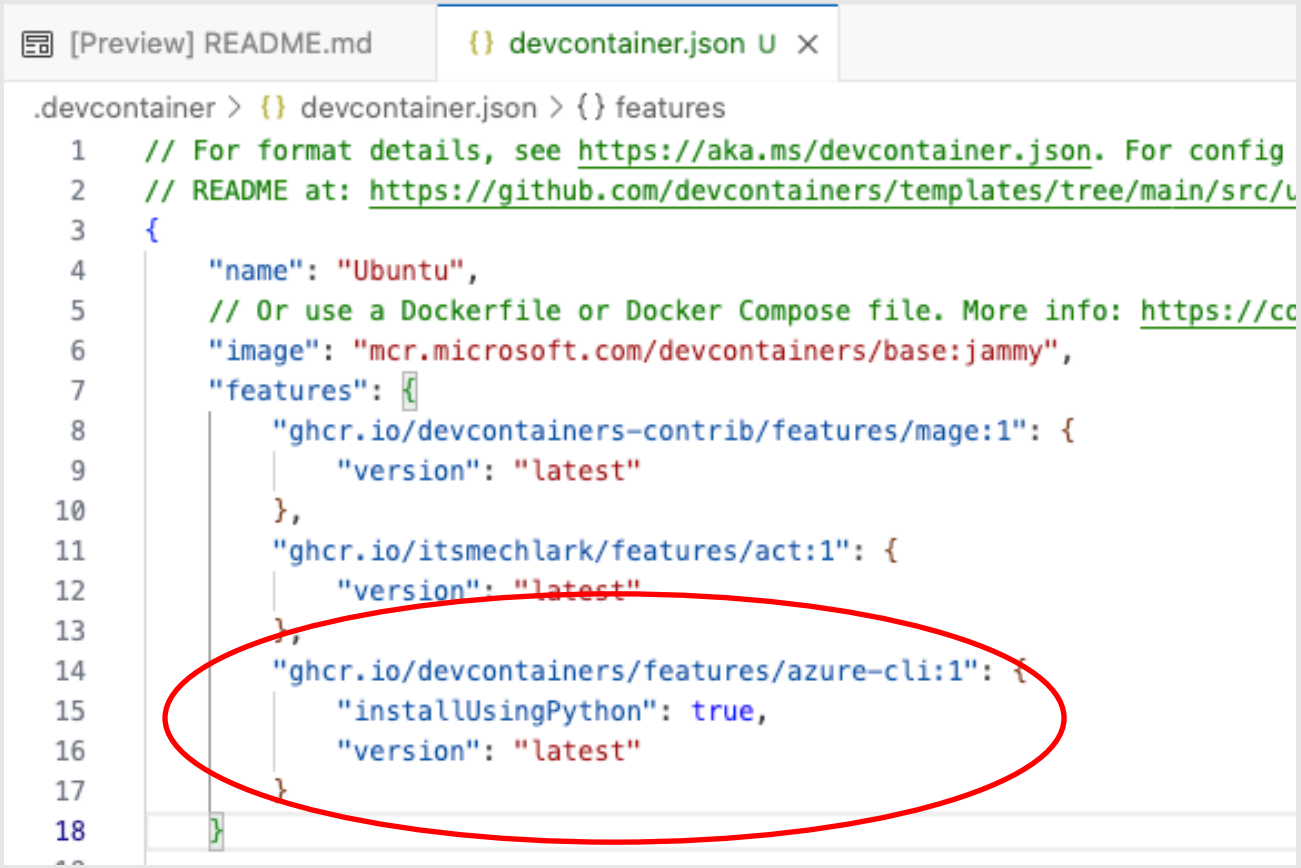


Common Utilities	Dev Container Spec Maintainers	ghcr.io/devcontainers/features/common-utils:2	2.1.1
Conda	Dev Container Spec Maintainers	ghcr.io/devcontainers/features/conda:1	1.0.9
Light-weight Desktop	Dev Container Spec Maintainers	ghcr.io/devcontainers/features/desktop-lite:1	1.0.8
Docker (Docker-in-Docker)	Dev Container Spec Maintainers	ghcr.io/devcontainers/features/docker-in-docker:2	2.4.0
Docker (docker-outside-of-docker)	Dev Container Spec Maintainers	ghcr.io/devcontainers/features/docker-outside-of-docker:1	1.3.0
Dotnet CLI	Dev Container Spec Maintainers	ghcr.io/devcontainers/features/dotnet:1	1.1.4

Adding features

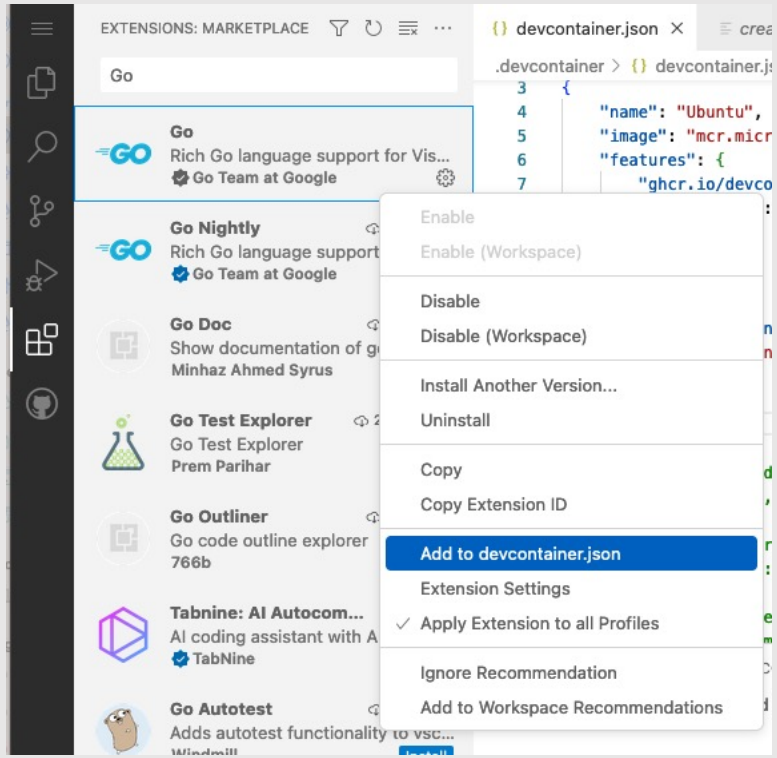
- "add dev"
- modify active config
- select feature
- choose defaults/options
- rebuild





Items in devcontainer.json - extensions

- Just like extensions in VS Code
- Are installed and run and have full access to tools, platform, and file system
- Structure is "customizations"->"vscode"-> "extensions"

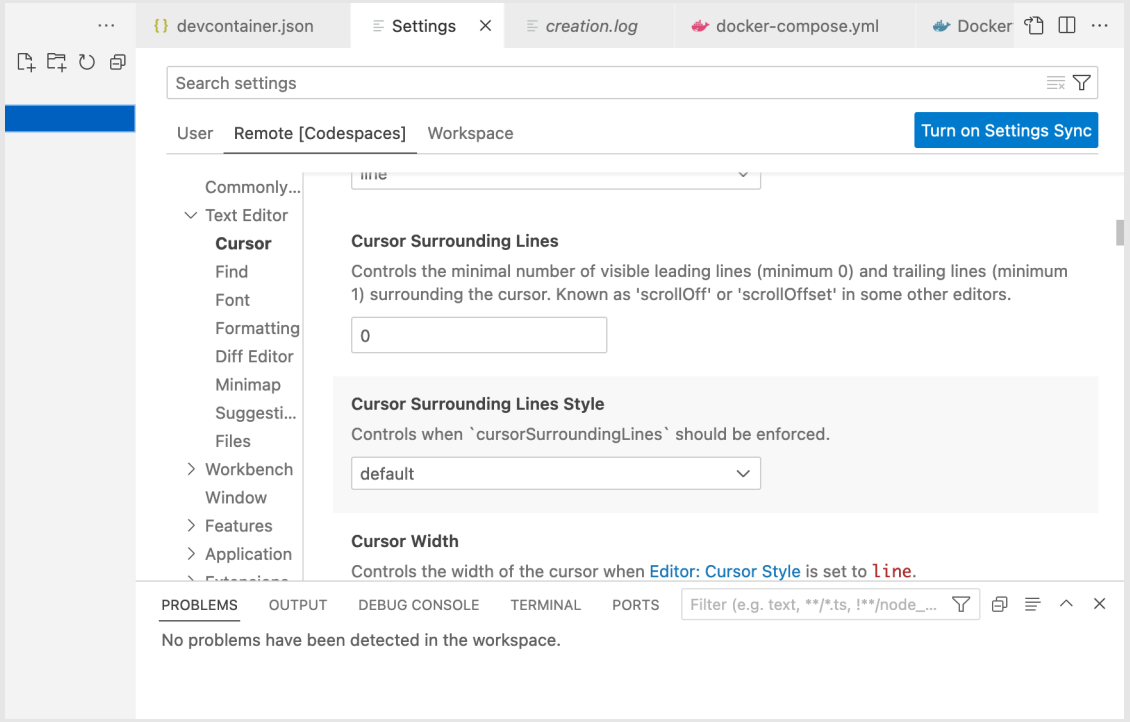


- Can add directly in code for dev container
- Can also add automatically insert code via
  - Open extensions in Activity Bar
  - Find the extension
  - Right click
  - Select on "Add to devcontainer.json"

Items in devcontainer.json - settings

- settings for environment
- similar to setting values in VS Code ~/.config/Code/User/settings.json
- Structure is "customizations"->"vscode"-> "extensions"
- Can get to settings via Ctrl/Cmd + ,
- Can activate "Settings Sync" to share your Visual Studio Code configurations
- Sync shares settings, keybindings, and installed extensions across your machines
- Allow you to always work with your favorite setup

```
"customizations": {
    "vscode": {
        "extensions": [
            "golang.go"
        ],
        "settings": {
            "terminal.integrated.shell.linux": "/bin/bash"
        }
    }
}
```



Dev Container Dockerfile option

- Can have a Dockerfile in .devcontainer folder and referenced in devcontainer.json
  - Alternative to Dockerfile is using "image" property
- To use Dockerfile as part of container, use "dockerfile" property in devcontainer.json

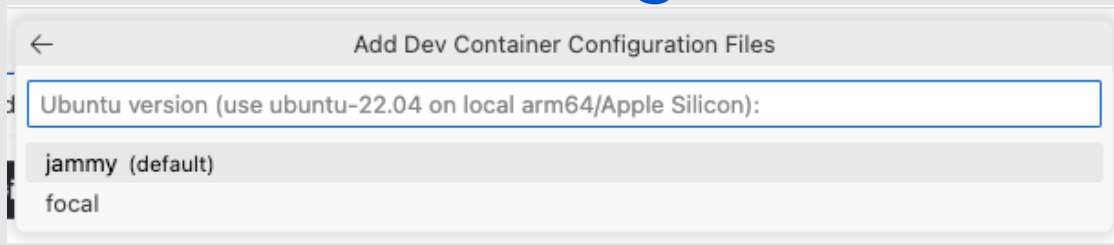
```
ARG VARIANT="16-buster"
FROM mcr.microsoft.com/vscode/devcontainers/javascript-node:0-${VARIANT}

# [Optional] Uncomment if you want to install an additional version of node using
nvm
# ARG EXTRA_NODE_VERSION=10
# RUN su node -c "source /usr/local/share/nvm/nvm.sh && nvm install
${EXTRA_NODE_VERSION}"

# [Optional] Uncomment if you want to install more global node modules
# RUN su node -c "npm install -g <your-package-list-here>"

COPY library-scripts/github-debian.sh /tmp/library-scripts/
RUN apt-get update && bash /tmp/library-scripts/github-debian.sh
```

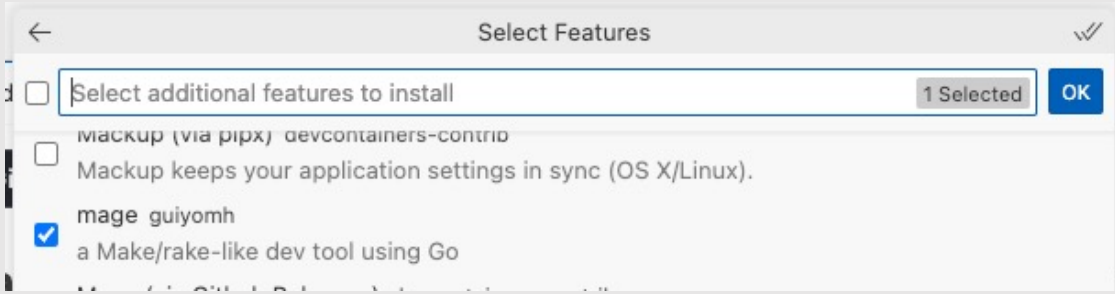
```
{
  // ...
  "build": { "dockerfile": "Dockerfile" },
  // ...
}
```

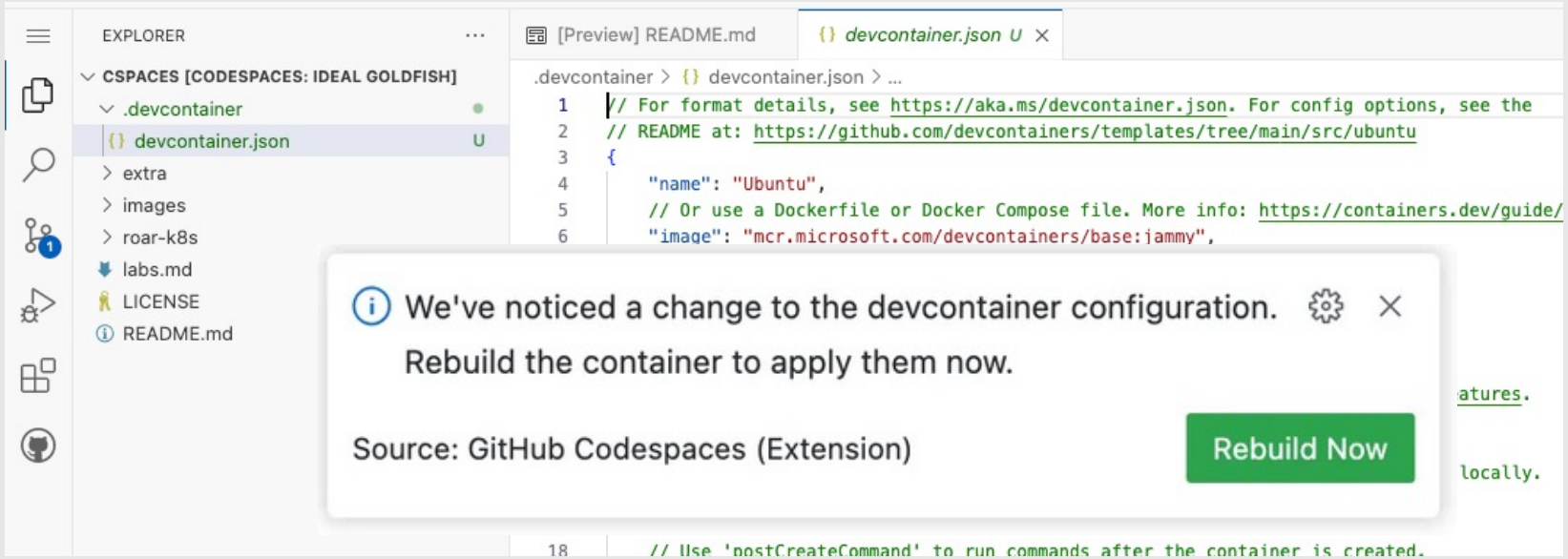
credit: https://docs.github.com/en/codespaces/setting-up-your-project-for-codespaces/adding-a-dev-container-configuration/introduction-to-dev-containers

Using a predefined dev container config

- Create a new configuration
- Show All Definitions
- Select Definition
- Follow prompts to customize
- Rebuild Now

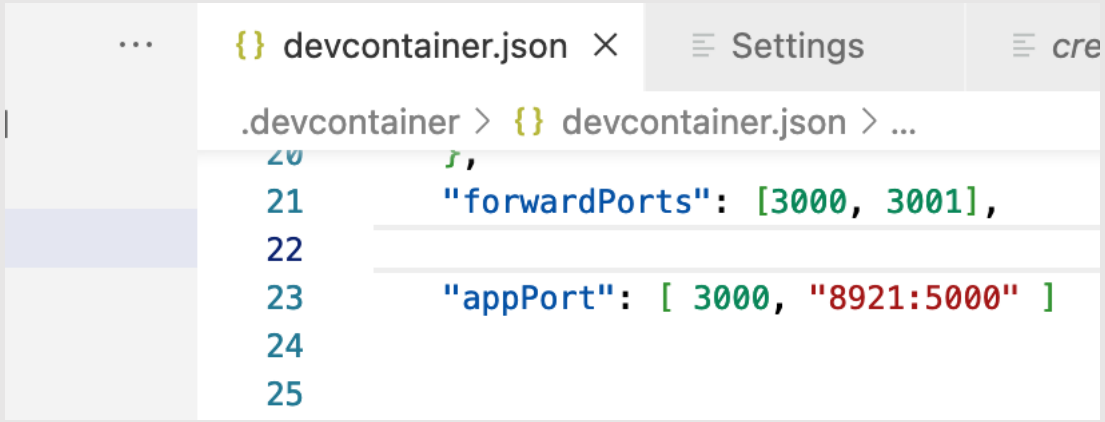






Items in devcontainer.json - ports

- Forwarding ports
  - Specify a list of ports you **always** want to forward when attaching or opening a folder in container
  - Look like localhost to the application.
  - "forwardPorts" property
- Publishing ports
  - Published ports behave very much like ports you make available to your local network.
  - If your application only accepts calls from localhost, it will reject connections from published ports just as your local machine would for network calls.
  - "appPort" property



Items in devcontainer.json - "post" commands

- When you rebuild your dev container, you can lose anything you've installed manually. To help with that, you can use the "postCreateCommand"
- Any "postCreateCommand" actions are run once the container is created, so you can also use the property to run commands like npm install or to execute a shell script in your source tree (if you have mounted it).

```
"postCreateCommand": "bash scripts/install-dev-tools.sh"
```

- You can also use an interactive bash shell so that your .bashrc is picked up, automatically customizing your shell for your environment:

```
"postCreateCommand": "bash -i scripts/install-dev-tools.sh"
```

- Tools like NVM won't work without using -i to put the shell in interactive mode:

```
"postCreateCommand": "bash -i -c 'nvm install --lts'"
```

. The command needs to exit or the container won't start. For instance, if you add an application start to postCreateCommand, the command wouldn't exit

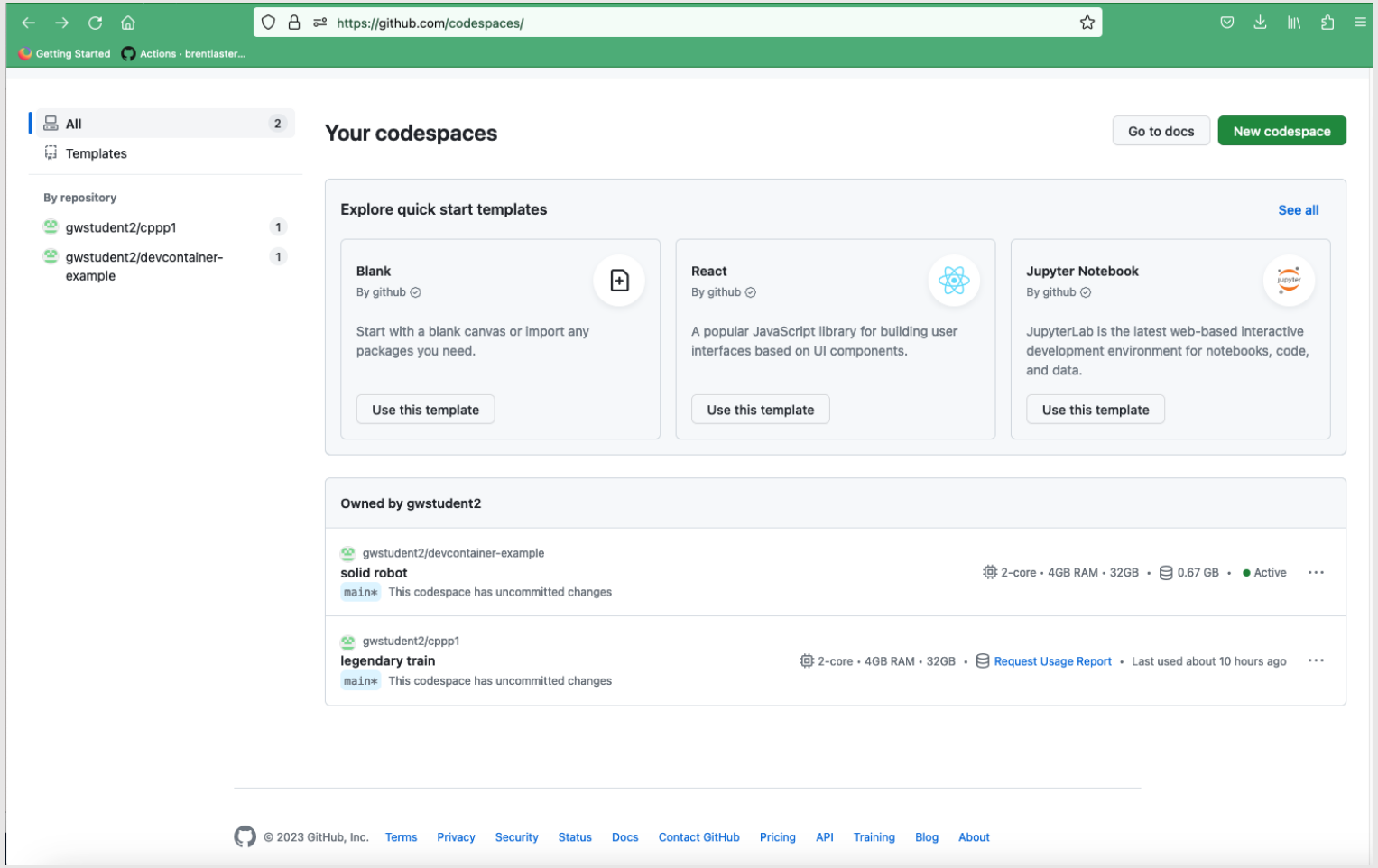
- Also have a "postStartCommand" that executes every time the container starts. The parameters behave exactly like postCreateCommand, but the commands execute on start rather than create.

Common components of devcontainer.json

Item	Purpose
name	Specifies the name of your Codespace
dockerFile	Points to your Dockerfile, which you need to create in the same directory as the devcontainer.json file
extensions	Lists any VS Code extensions you want to install in your Codespace
settings	Defines VS Code settings specific to your Codespace
forwardPorts	Specifies the port(s) to forward from the container to your local machine
postCreateCommand	Specifies the command to run after the container is created
workspaceMount	Mounts your local workspace folder into the container
workspaceFolder	Defines the folder inside the container where the workspace will be mounted
remoteUser	Specifies the user to use inside the container
features	Adds language/tool/CLI to a dev container

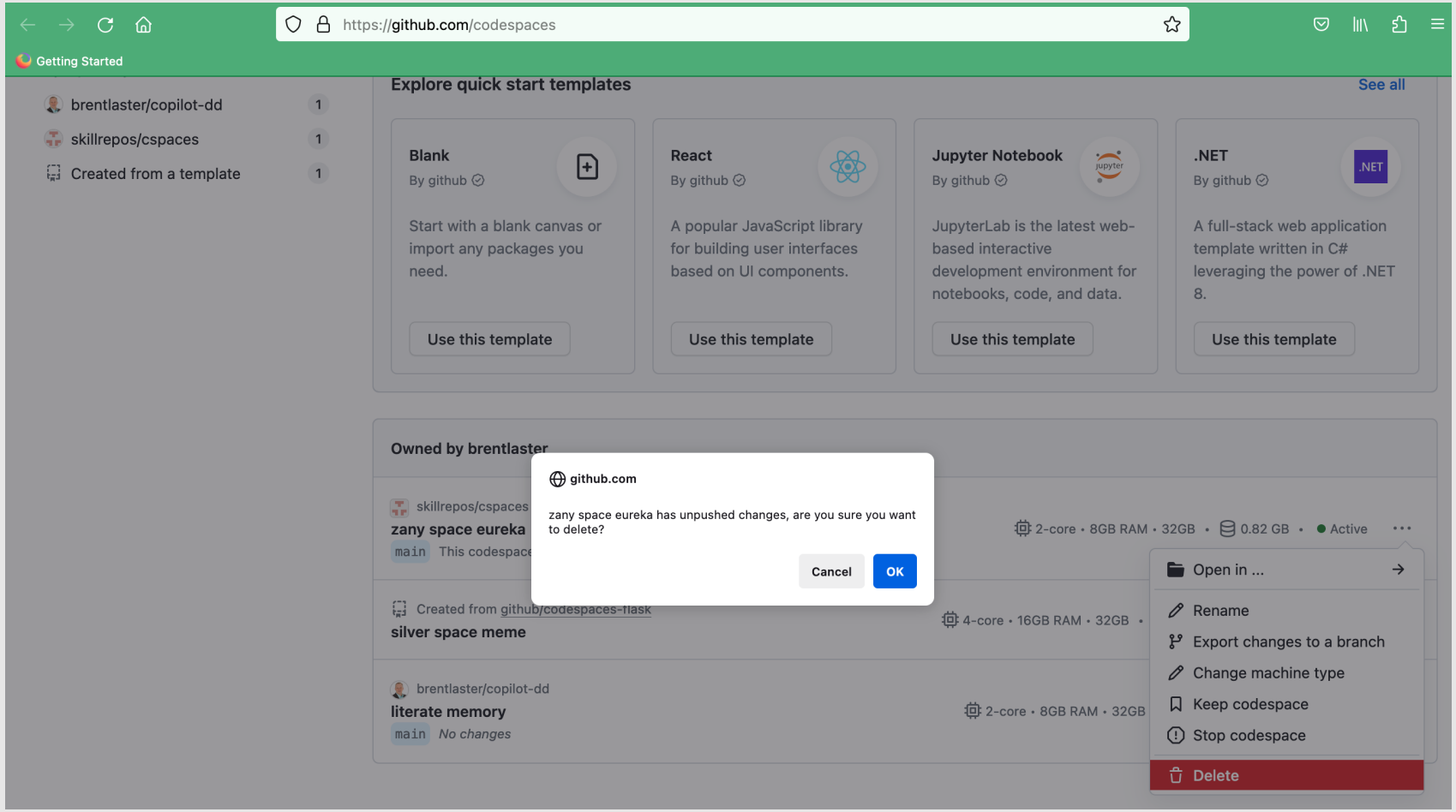
Seeing your codespaces

- When logged in, go to github.com/codespaces



Deleting a codespace

- Click the menu and select "Delete"

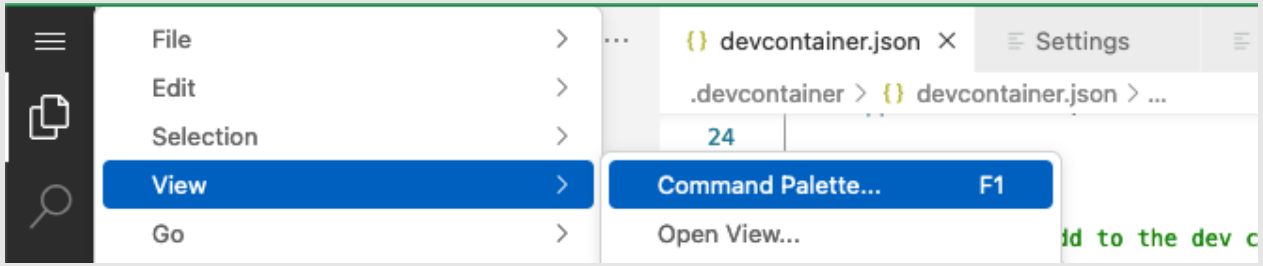


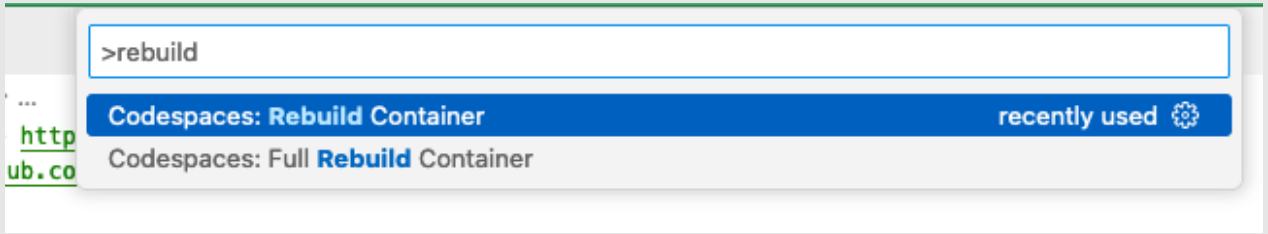
Lab 3 – Creating Dev Containers

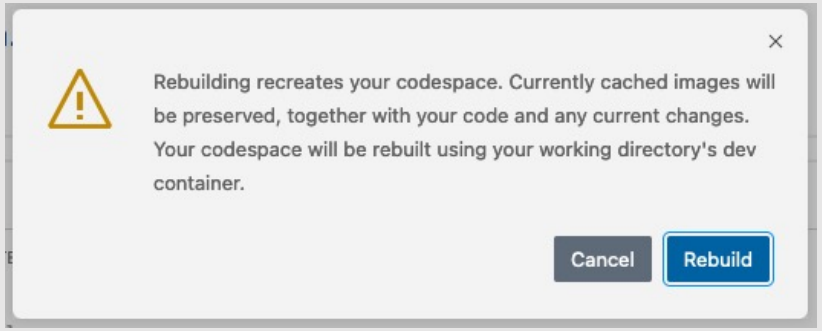
Purpose: In this lab, we'll see how to create and work with a dev container to persist changes across codespaces.

Rebuilding a container

- May need to rebuild a container to update it with configuration changes in codespace
- In codespace, dev env is a Docker container that executes on a VM
- To pick up changes, rebuild
- If possible, Codespaces will use cached images from previous builds
- Full rebuild option cleans all images, containers, and volumes from cache, then redoes everything
- To rebuild a container
  - Bring up the Command Palette (Shift + Cmd + P / Ctrl + Shift + P) or F1 or View->Palette
  - Type "rebuild" in Command Palette and select option
- GitHub CLI can also be used to rebuild
  - In terminal: `gh codespace rebuild`
  - `Use arrow keys to select codespace`
  - `For full rebuild, use $gh codespace rebuild --full`



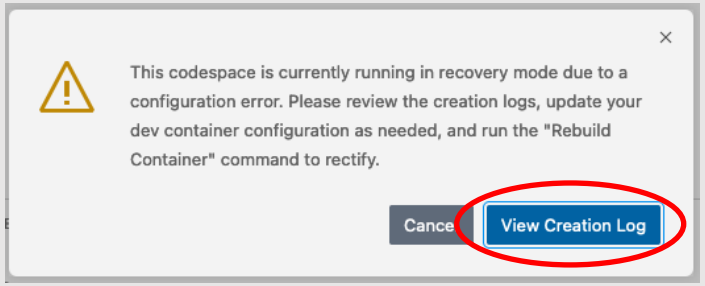


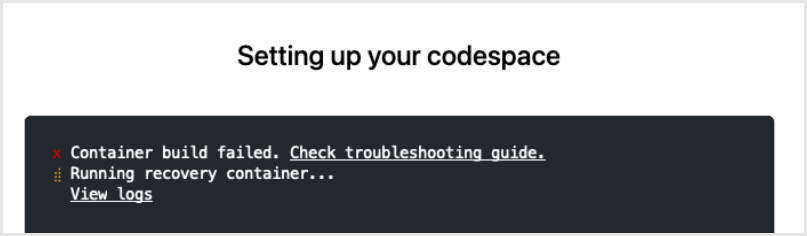


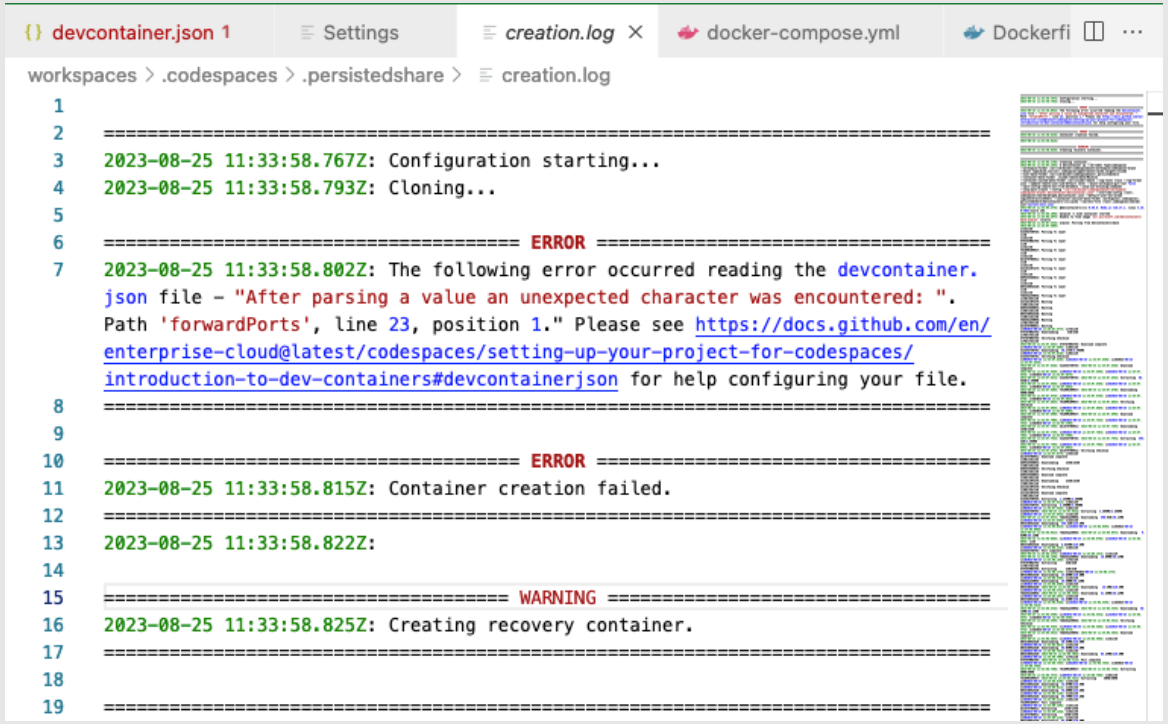

```
@gwstudent2 → /workspaces/codespaces-blank $ gh codespace rebuild
? Choose codespace:  [Use arrows to move, type to filter]
> github/codespaces-blank (main): bug-free potato
```

Failed container builds

- If container build fails, first indication is in "setting up" window
- After startup, error dialog
- Then can choose to view Creation Log
- Errors also visible in PROBLEMS tab

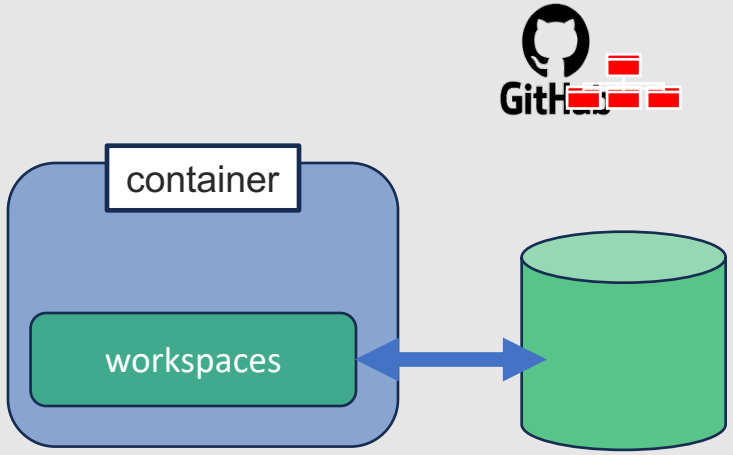






Persisting data across rebuilds

- When codespace is created, /workspaces is mounted into container as persistent directory
- GitHub repo is cloned into /workspaces/
- Any changes made inside this directory are preserved on codespace starts, stops, rebuilds
- Includes any editing, adding, deleting of files in /workspaces
- Separate from /workspaces, codespace contains a Linux dir structure dependent on dev container used
- Other items preserved on start and stop
- To preserve files outside of /workspaces over a rebuild, create a directory and symlink into /workspaces
- Could even put shell script as postCreateCommand in devcontainer.json file



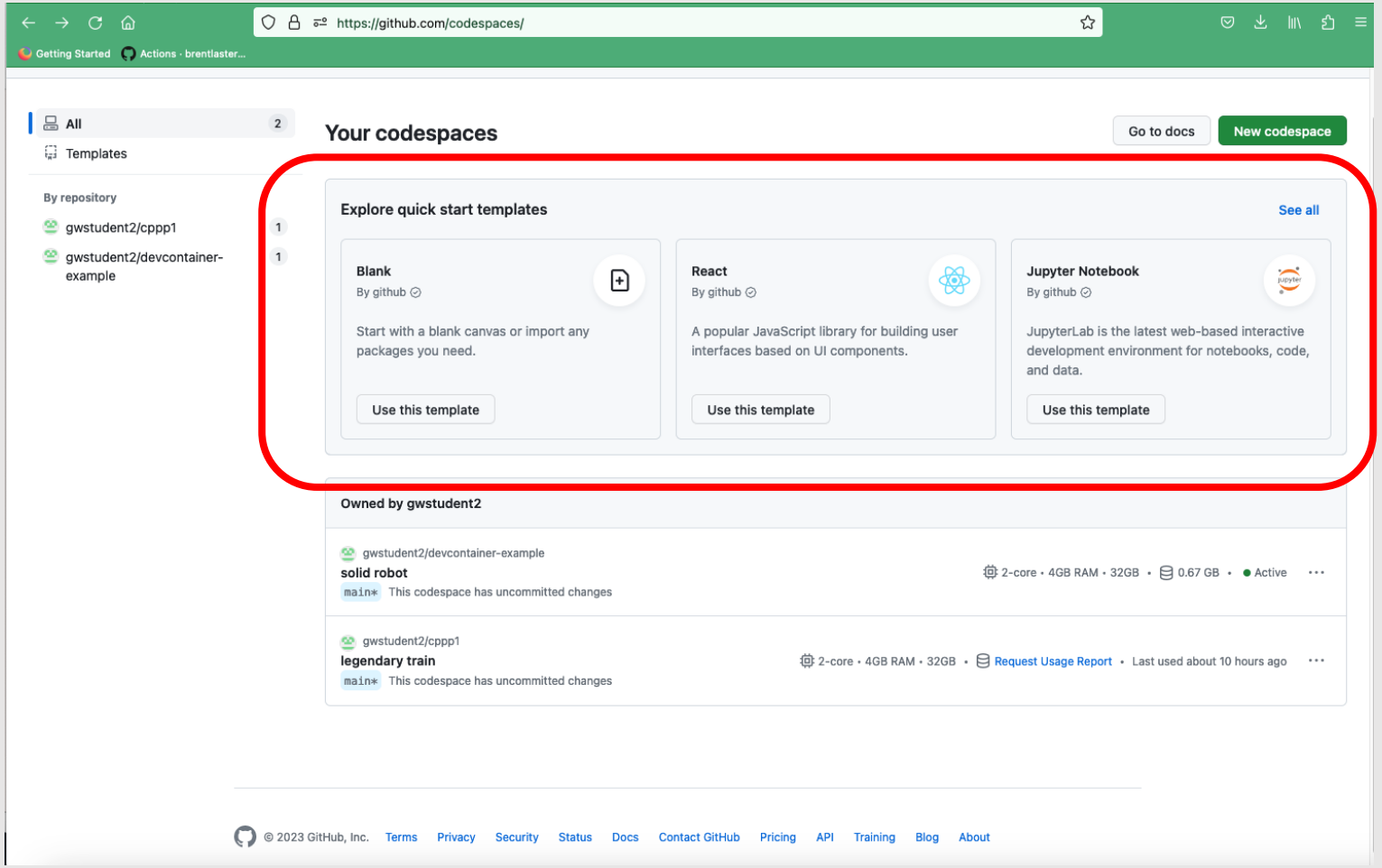

```
#!/bin/bash
ln -sf $PWD/.devcontainer/config $HOME/config && set +x
```

```
{
    "image": "mcr.microsoft.com/vscode/devcontainers/base:alpine",
    "postCreateCommand": ".devcontainer/postCreate.sh"
}
```

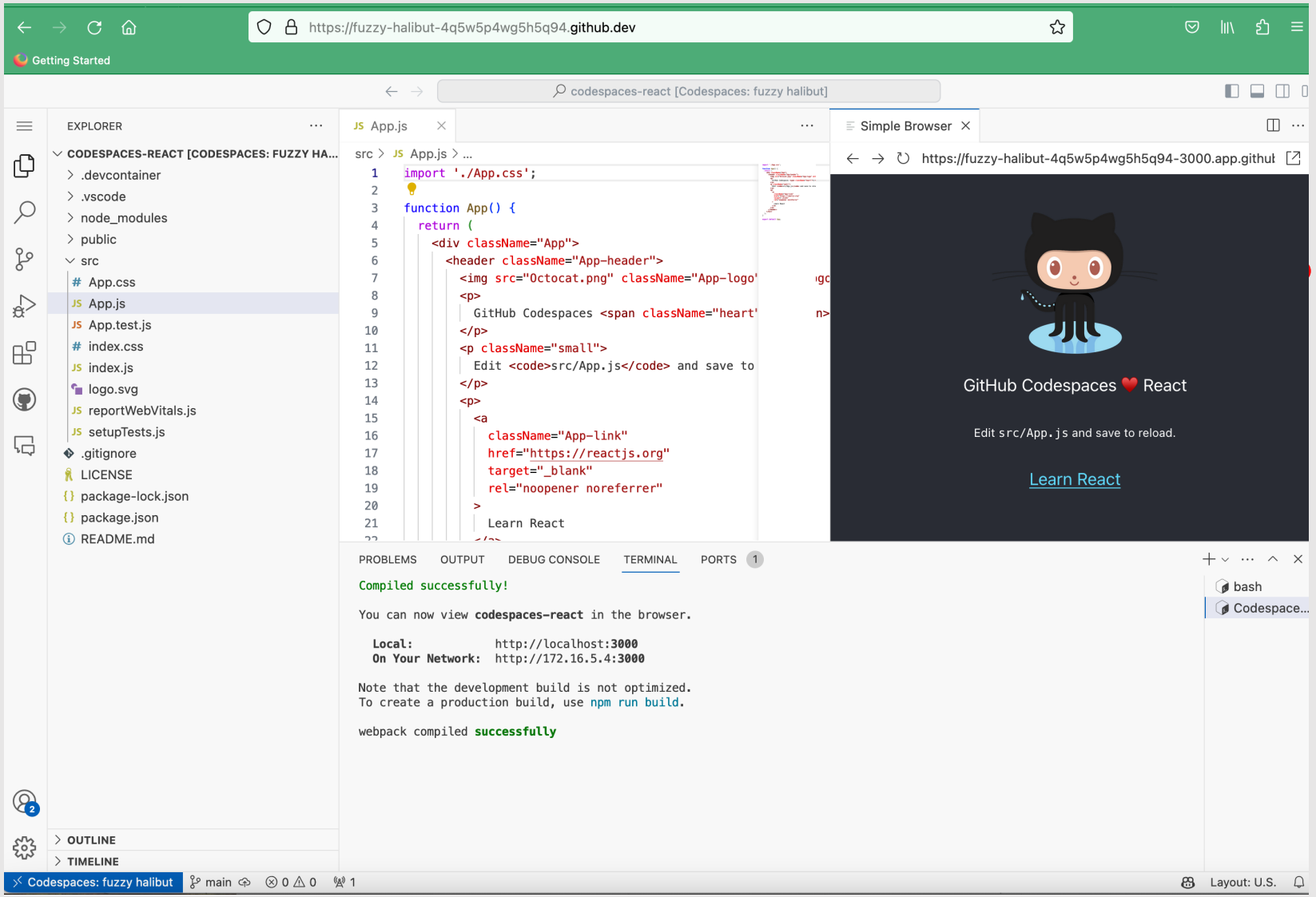
Credit: https://docs.github.com/en/codespaces/developing-in-a-codespace/rebuilding-the-container-in-a-codespace#persisting-data-over-a-rebuild

Seeing codespace templates

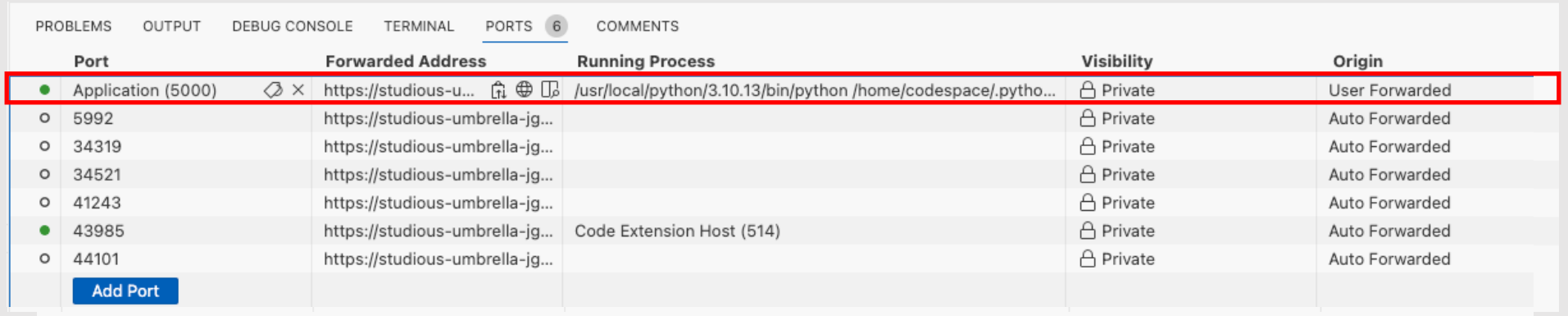
- When logged in, go to github.com/codespaces



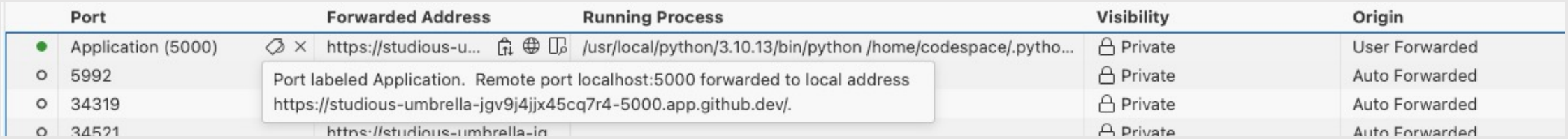
Using a template to create a codespace



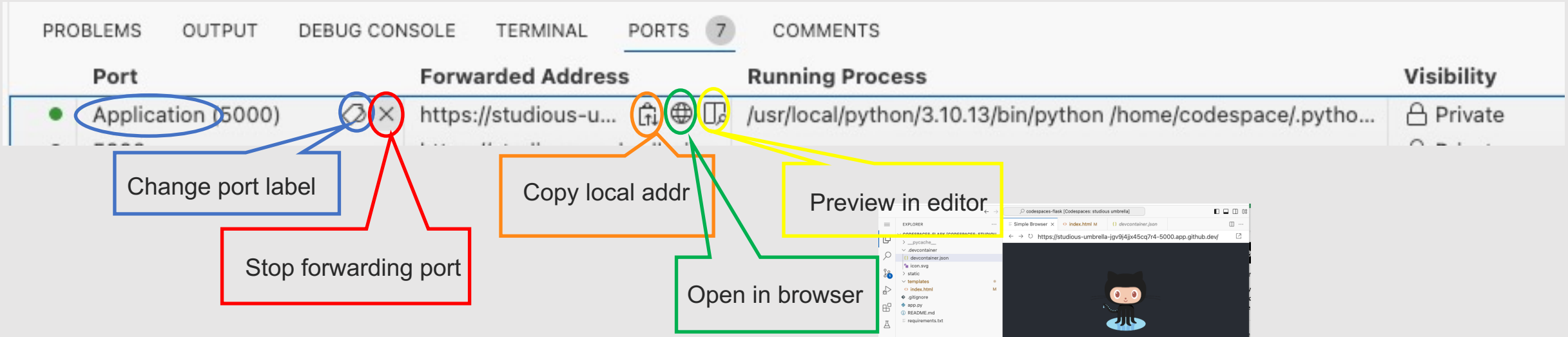
Forwarding a port

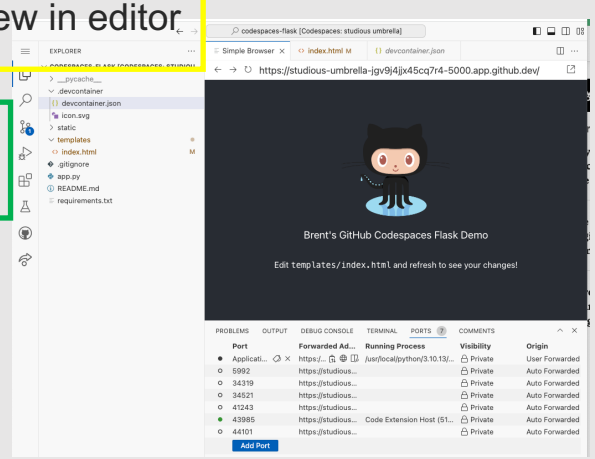


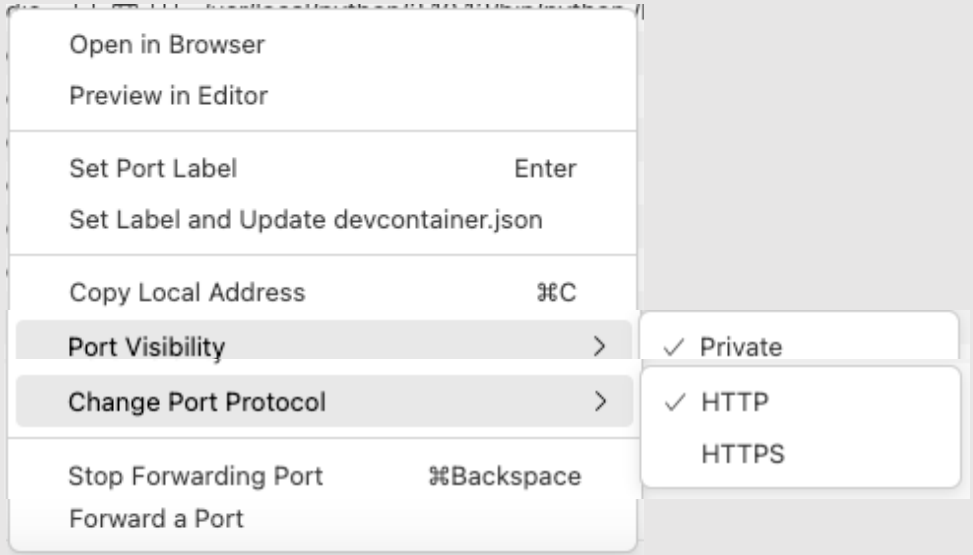
- PORTS tab
- Add Port button
- If known port will be added with relevant data



Working with forwarded ports

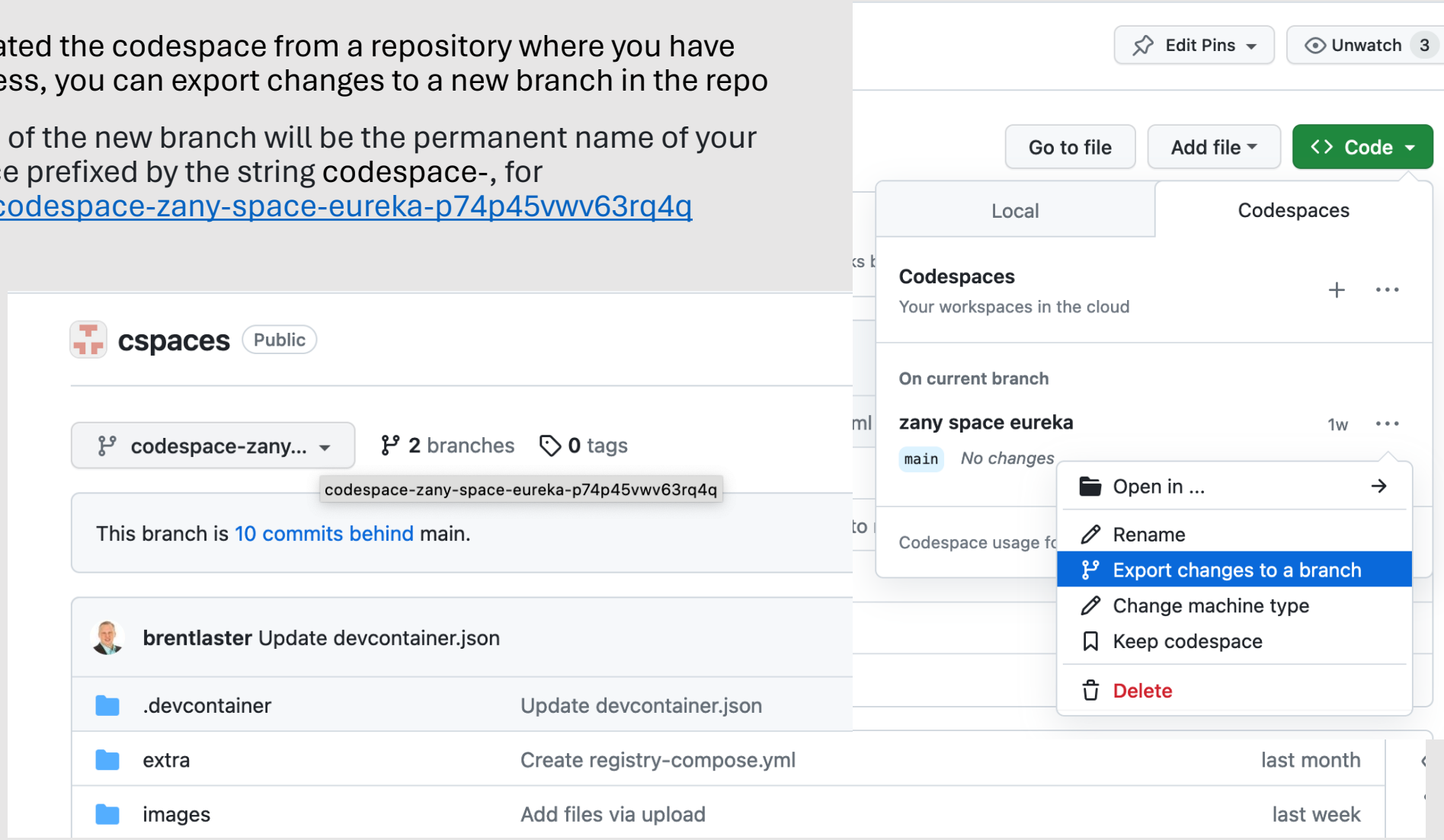






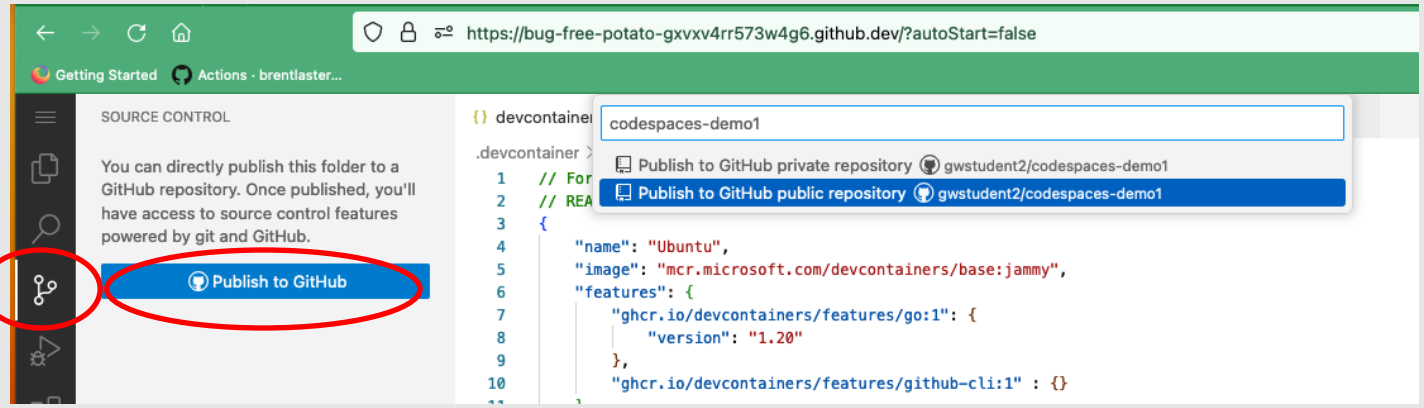
Exporting codespaces to a branch

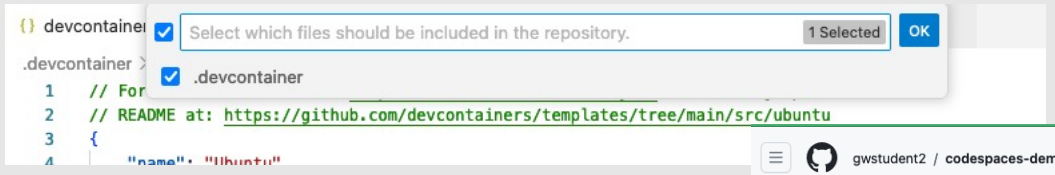
- If you created the codespace from a repository where you have write access, you can export changes to a new branch in the repo
- The name of the new branch will be the permanent name of your codespace prefixed by the string codespace-, for example [codespace-zany-space-eureka-p74p45vwv63rq4q]

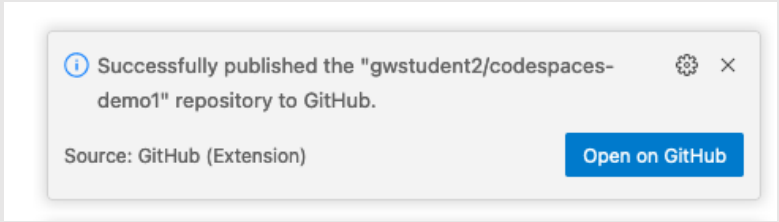


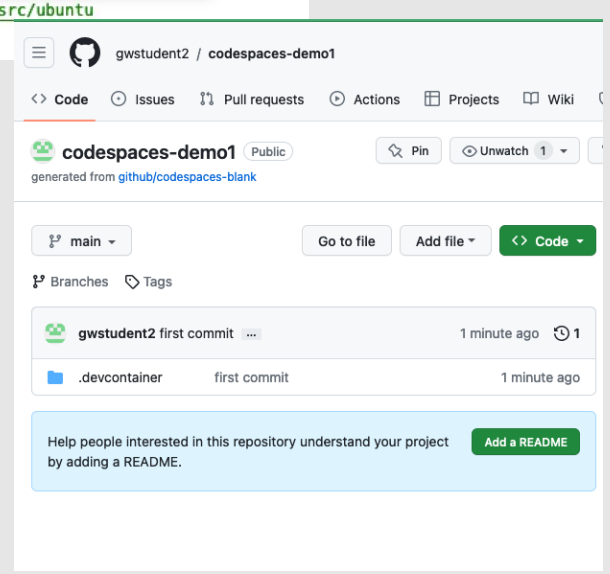
Codespaces publish to new repo

- If code is based **on a template** (not associated with existing repo...)
- Click on source control icon in Activity Bar
- Click on "Publish to GitHub" button
- Brings up command pallet with action
- Follow prompts







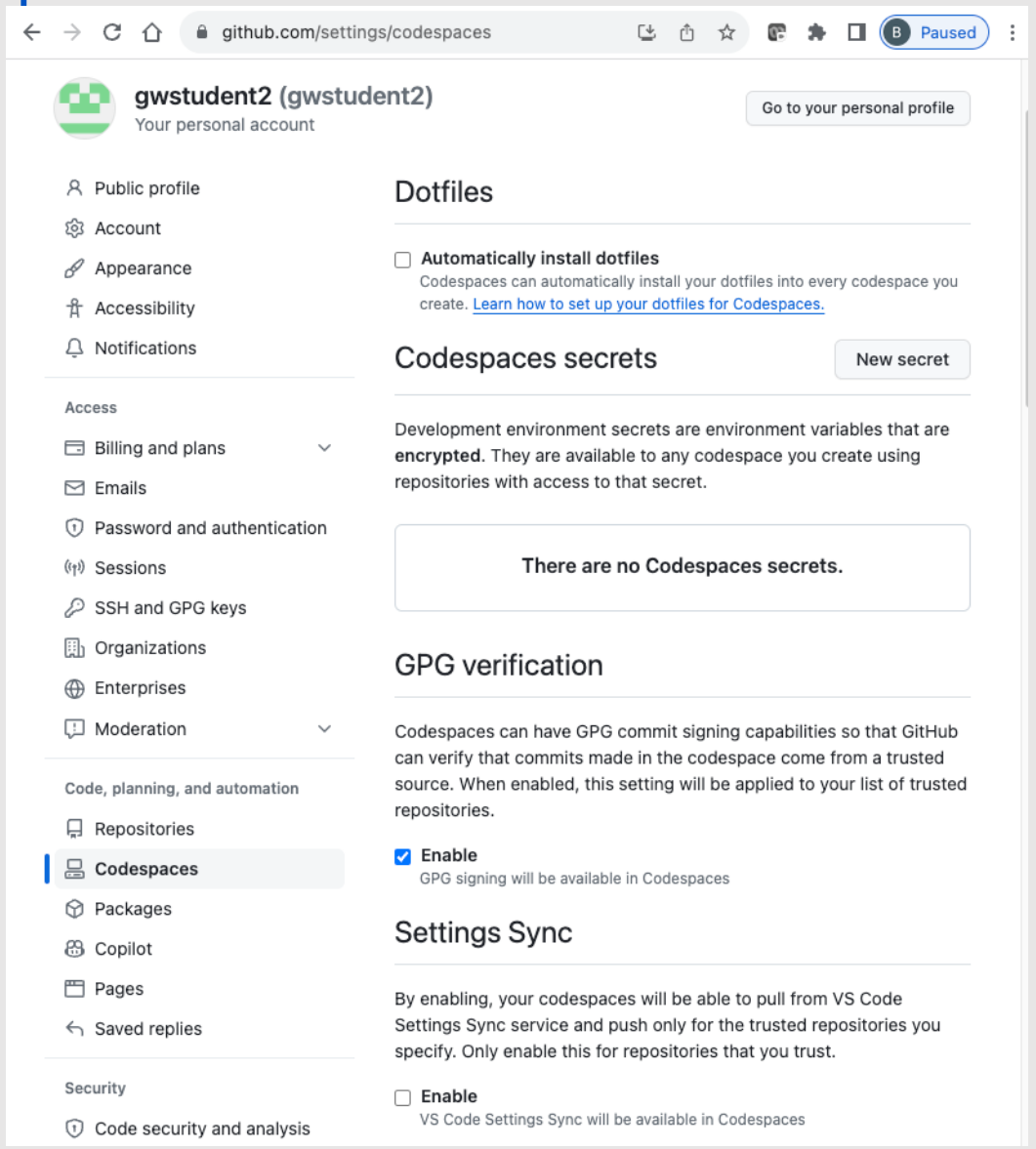


Lab 4 - Templates and Ports

Purpose: In this lab, we'll see how to create a codespace from a template and also work with ports.

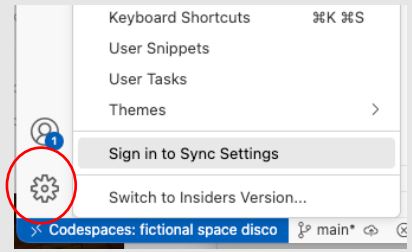
Personalizing GitHub Codespaces

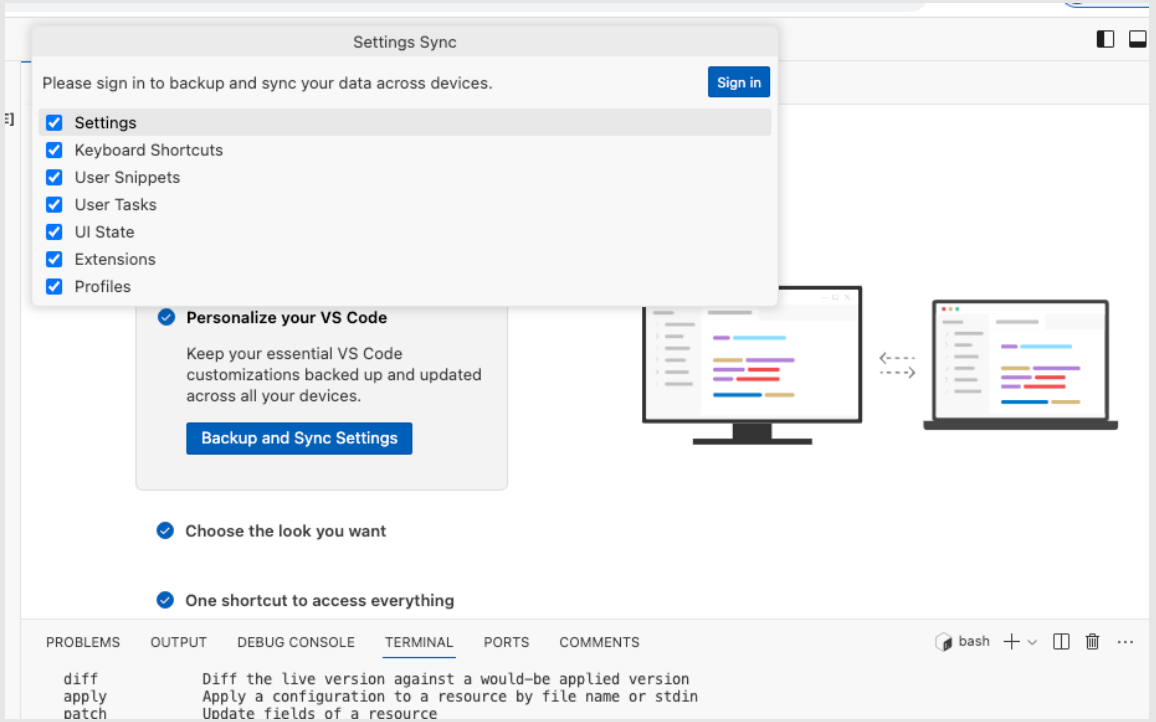
- Settings Sync - Synchronize your VS Code settings between the desktop app and VS Code web client
- Dotfiles - use a *dotfiles* repo to specify scripts, shell preferences, and other configurations
- Settings->Codespaces
- Defaults config that applies to every codespace for a repo

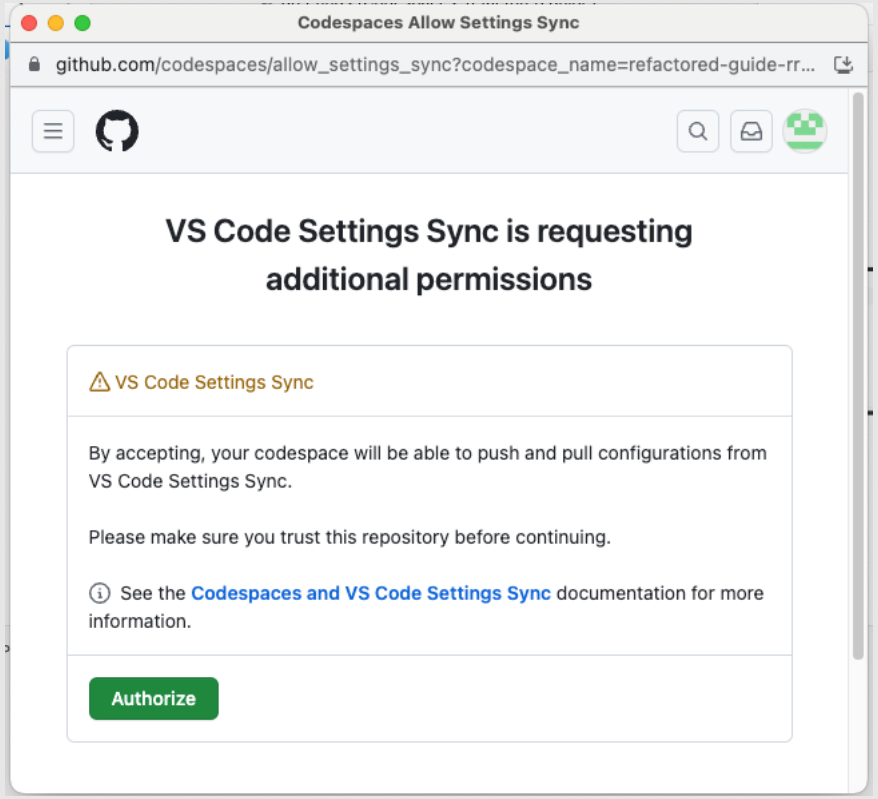


Settings sync

- Allows you to sync "look and feel" across interfaces
- Can select Manage (bottom icon) and sign in

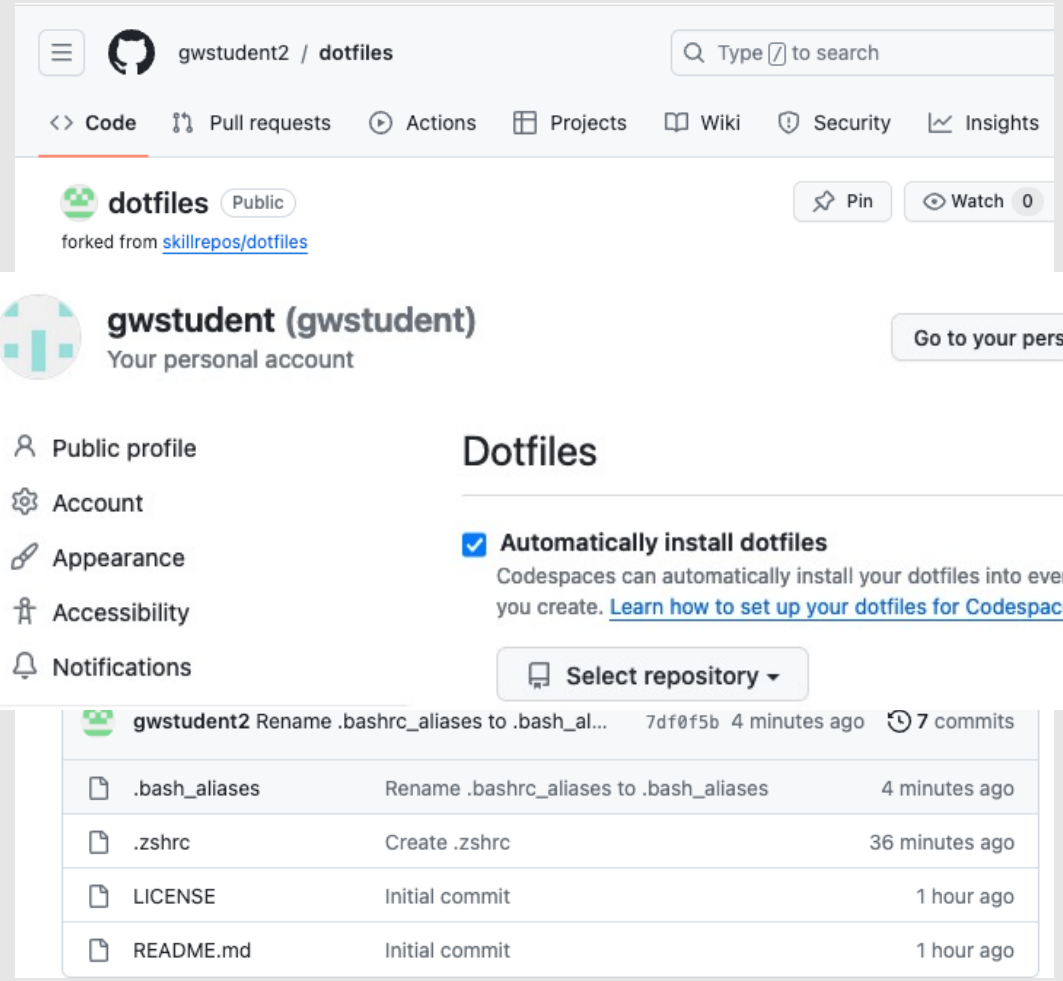


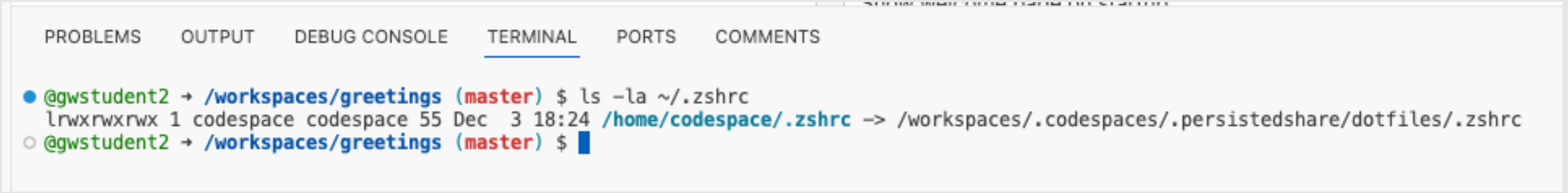




Dotfiles

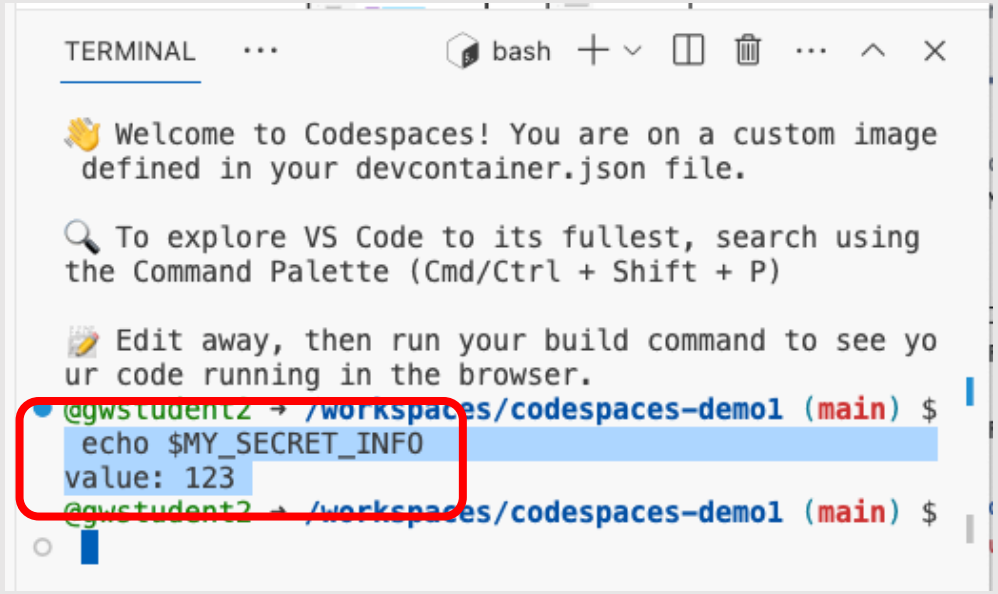
- Files and folders starting with . that control config
- Can store dotfiles in GitHub repo
- Point codespaces to dotfiles repo
  - https://github.com/settings/codespaces
- Codespace will get dotfile and clone them into repo
- Dotfiles repo looks for typical startup file to setup the env (***install.sh, install, bootstrap.sh, script/bootstrap, setup.sh setup, script/setup***)
- If found, runs one of those
- If not found, will symlink any files starting with "."
- NOTE: Must be "dotfiles" repo for current logged in user

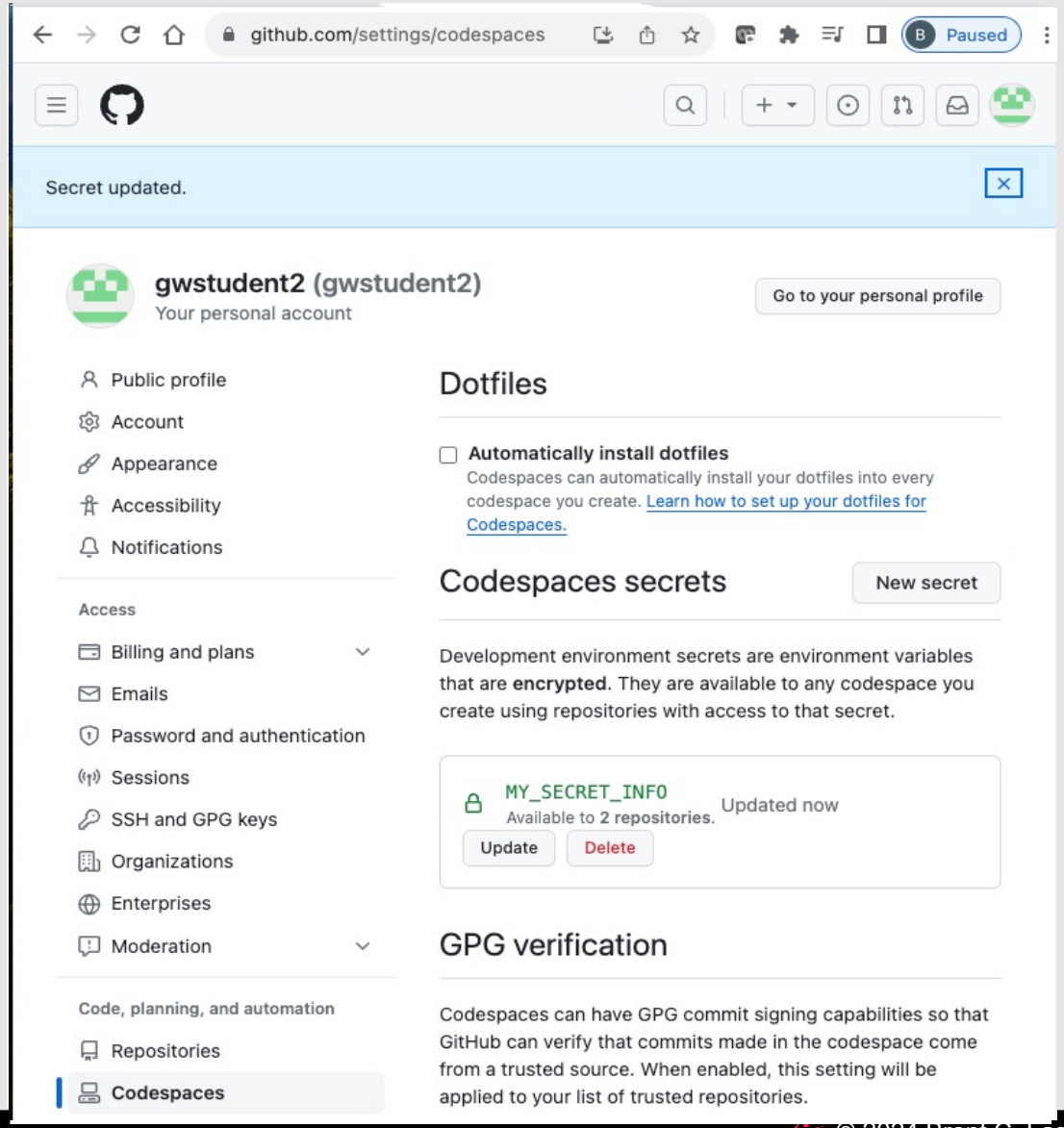




Codespace Secrets

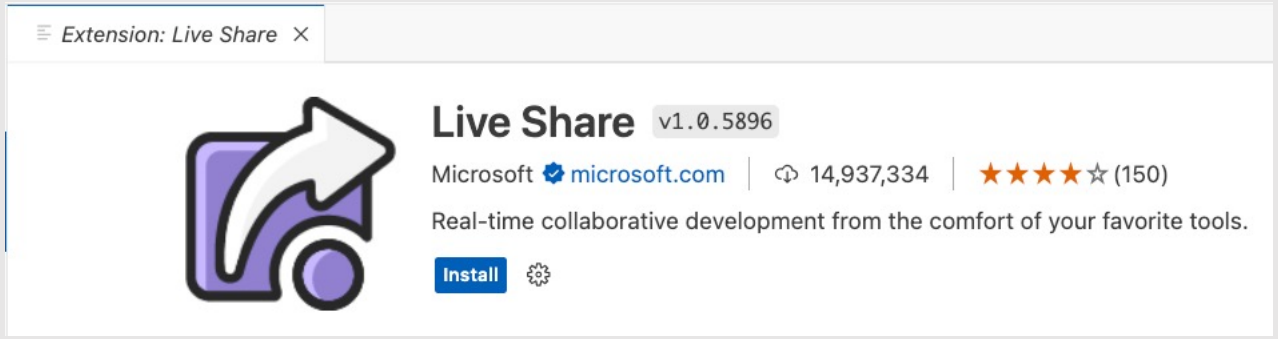
- Can create secrets specifically for codespaces
- Define which repos have access to the secrets (org, repo, etc.)
- Reference secrets as environment variables
- Secret values may be referenced in codespaces as env variables

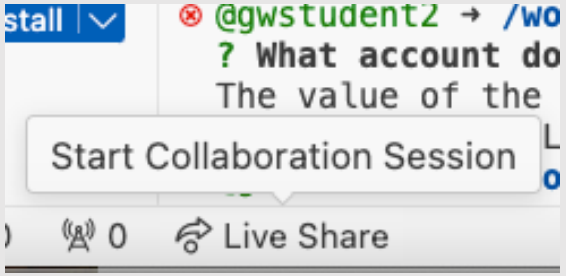


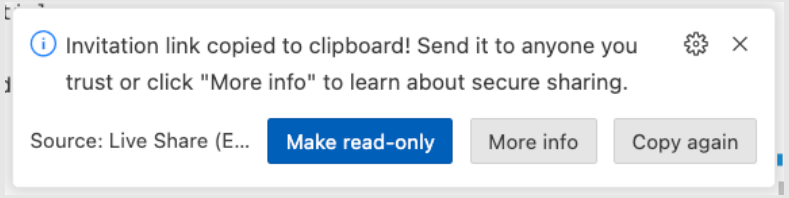


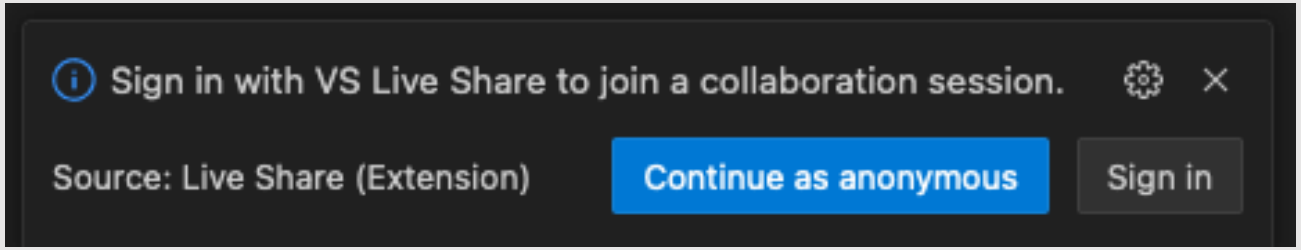
Live sharing

- Done via Live Share extension
- Allows for quick collaboration w/o needing to sync code or configure same dev environment
- Shared context of workspace in each editor
- Actions and changes from either participant are instantly reflected
- Can debug as well







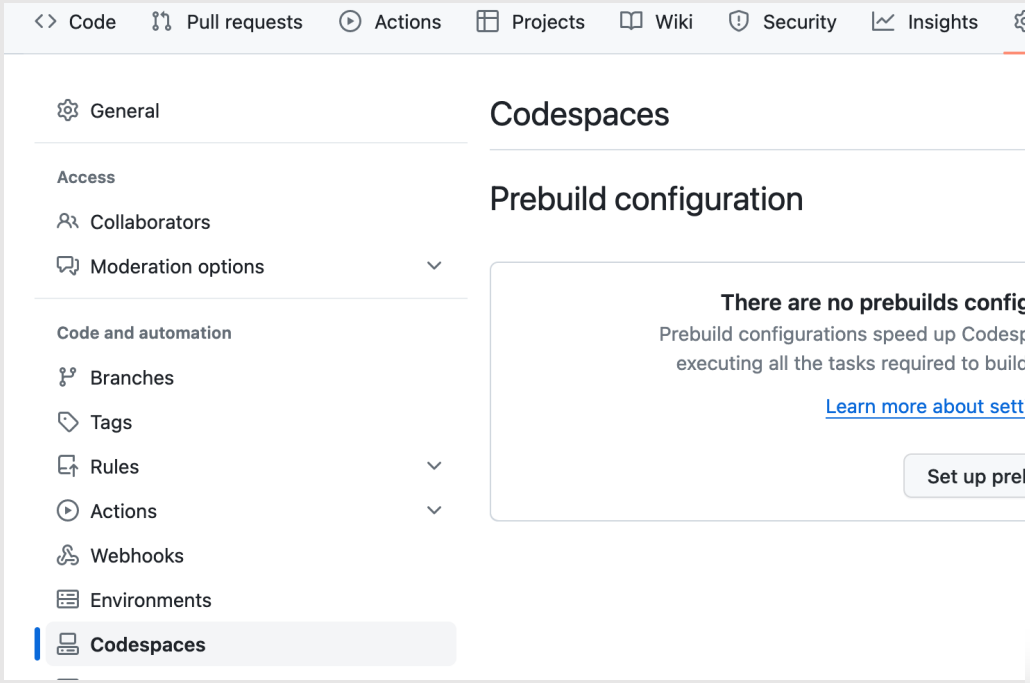


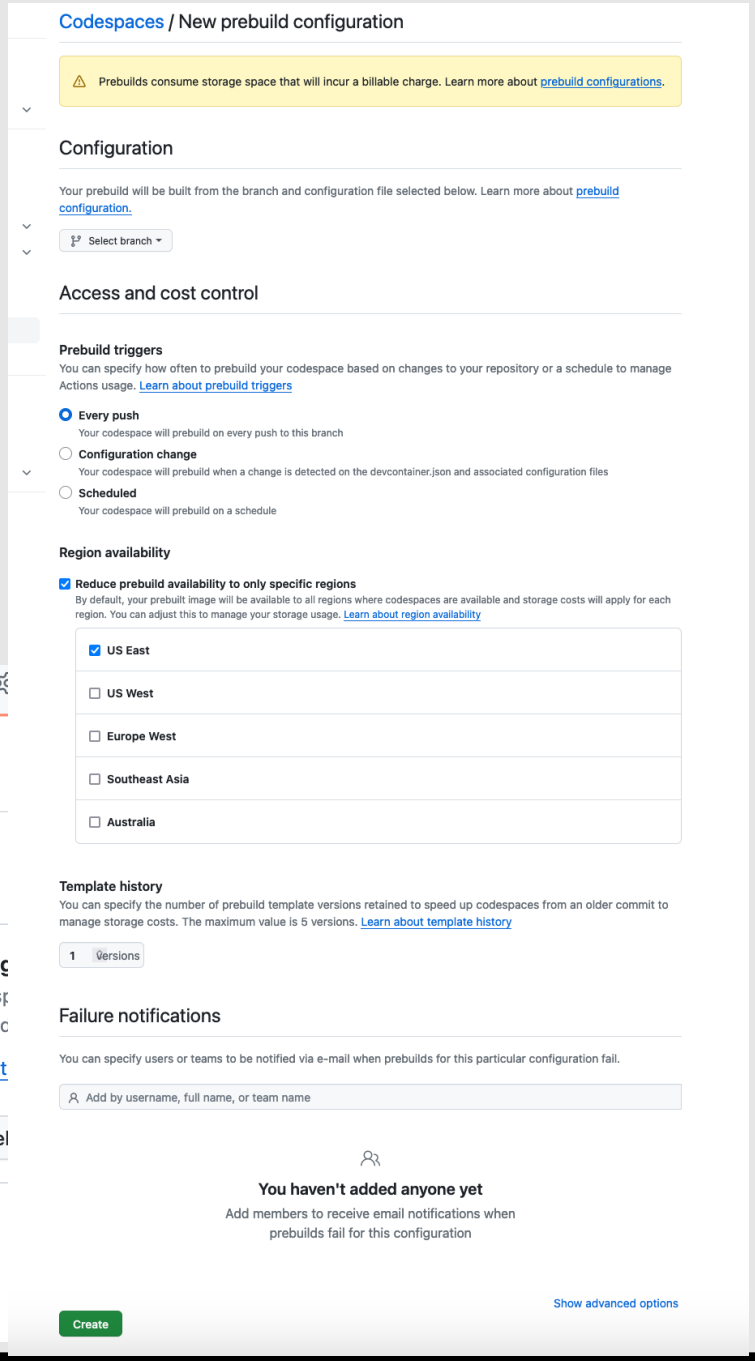
Lab 5 - Live sharing and collaboration

Purpose: In this lab, we'll see how to share a codespace and collaborate with it.

Codespace Prebuilds

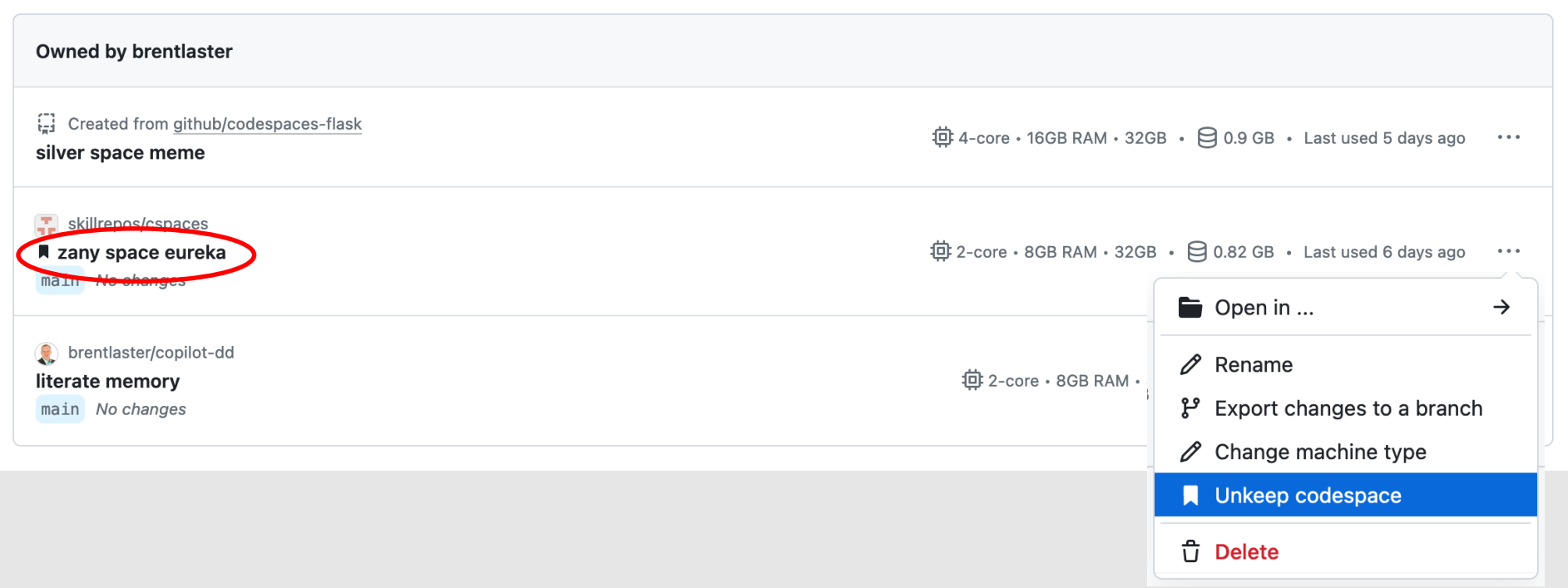
- Pre-build functionality
  - automatically pre-assembles pieces needed for codespaces for repo+branch+devcontainer.json
  - push to branch w/ prebuild runs GH Action to rebuild image
  - uses space to store prebuilt image that will be billable or come out of existing bucket





Checking expiration date

- If close to expiring, can see at github.com/codespaces
- Ones marked as keep have bookmark icon
- If marked as keep, there is an "unkeep" option



Managing codespaces with gh CLI

- GitHub CLI has "codespace" command
- commands such as *create, list, stop,* etc.
- Can select options via arrow keys if multiple (easier than full command line options)

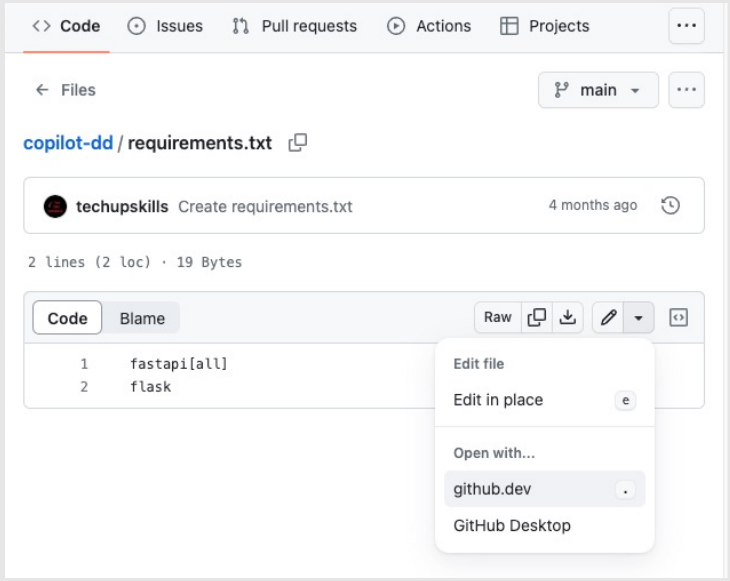
```
developer@Bs-MacBook-Pro calc3 % gh codespace
Connect to and manage codespaces

USAGE
  gh codespace [flags]

AVAILABLE COMMANDS
  code:        Open a codespace in Visual Studi
  cp:          Copy files between local and rem
  create:      Create a codespace
  delete:      Delete codespaces
  edit:        Edit a codespace
  jupyter:     Open a codespace in JupyterLab
  list:        List codespaces
  logs:        Access codespace logs
  ports:       List ports in a codespace
  rebuild:     Rebuild a codespace
  ssh:         SSH into a codespace
  stop:        Stop a running codespace
  view:        View details about a codespace
```

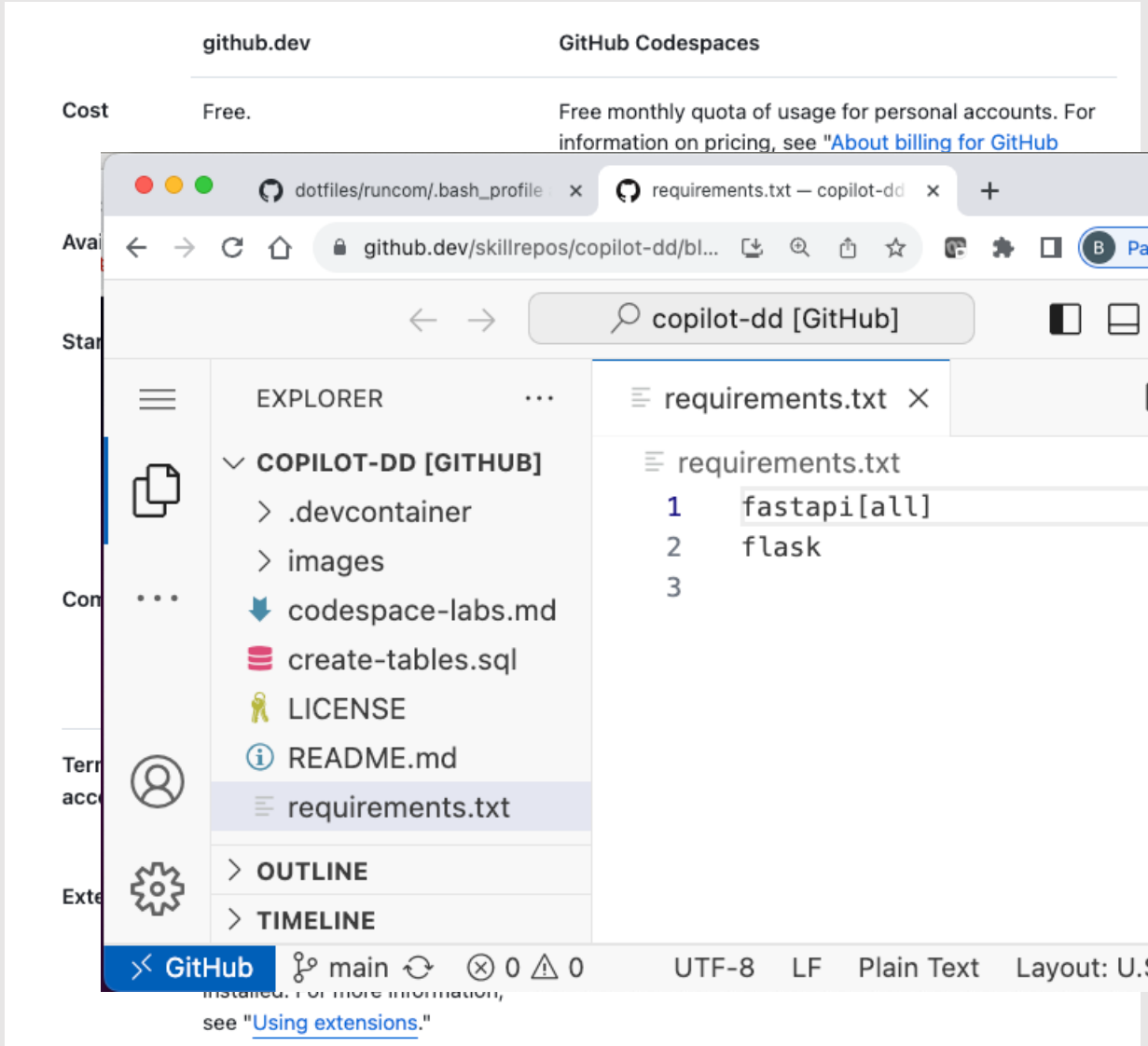
```
developer@Bs-MacBook-Pro calc3 % gh codespace list
NAME              DISPLAY NAME   REPOSITORY     BRANCH  STATE      CREATED AT
opulent-tribb...  opulent tr...  gwstudent2...  main*   Shutdown   about 3 da...
ideal-goldfis...  ideal gold...  skillrepos...  main*   Available  about 23 h...
```

```
developer@Bs-MacBook-Pro calc3 % gh codespace rebuild
? Choose codespace:  [Use arrows to move, type to filter]
> skillrepos/cspaces (main*): ideal goldfish
  gwstudent2/caz-class-v2 (main*): opulent tribble
```

GitHub.dev editor vs Codespaces

- GitHub.dev editor
  - lightweight editor that runs entirely in your browser
  - navigate files and repos
  - make and commit code changes
  - open any repo, fork, or PR in editor
  - Includes search syntax highlighting, and source control view





That's all - thanks!

Contact: training@getskillsnow.com

techskillstransformations.com
getskillsnow.com