



Dev Containers

Development environments as code

Alex Thissen

Xebia



Microsoft®
Most Valuable
Professional



10

YEARS

07.NOV.2024

DEV DAY

ELEVATING THE DEVELOPERS' COMMUNITY

Getting started with Dev Containers



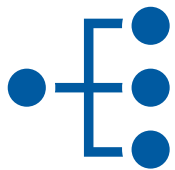
Dev environments and experiences

Preparation



Setup time

Prerequisites and dependencies
Many tools and frameworks



Maintain versions

Branches with different tools
Side-by-side install

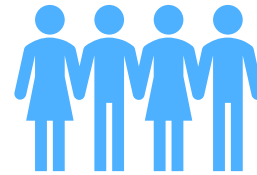


Remote work

Secure, safe and compliant
Any place in world



Team work



Differentiation

Different tools and requirements in
multidisciplinary team



Personalization

Your development environment
Personal preferences



Peer reviews

Pull requests
Clone (remote) repositories

Demo

Containers



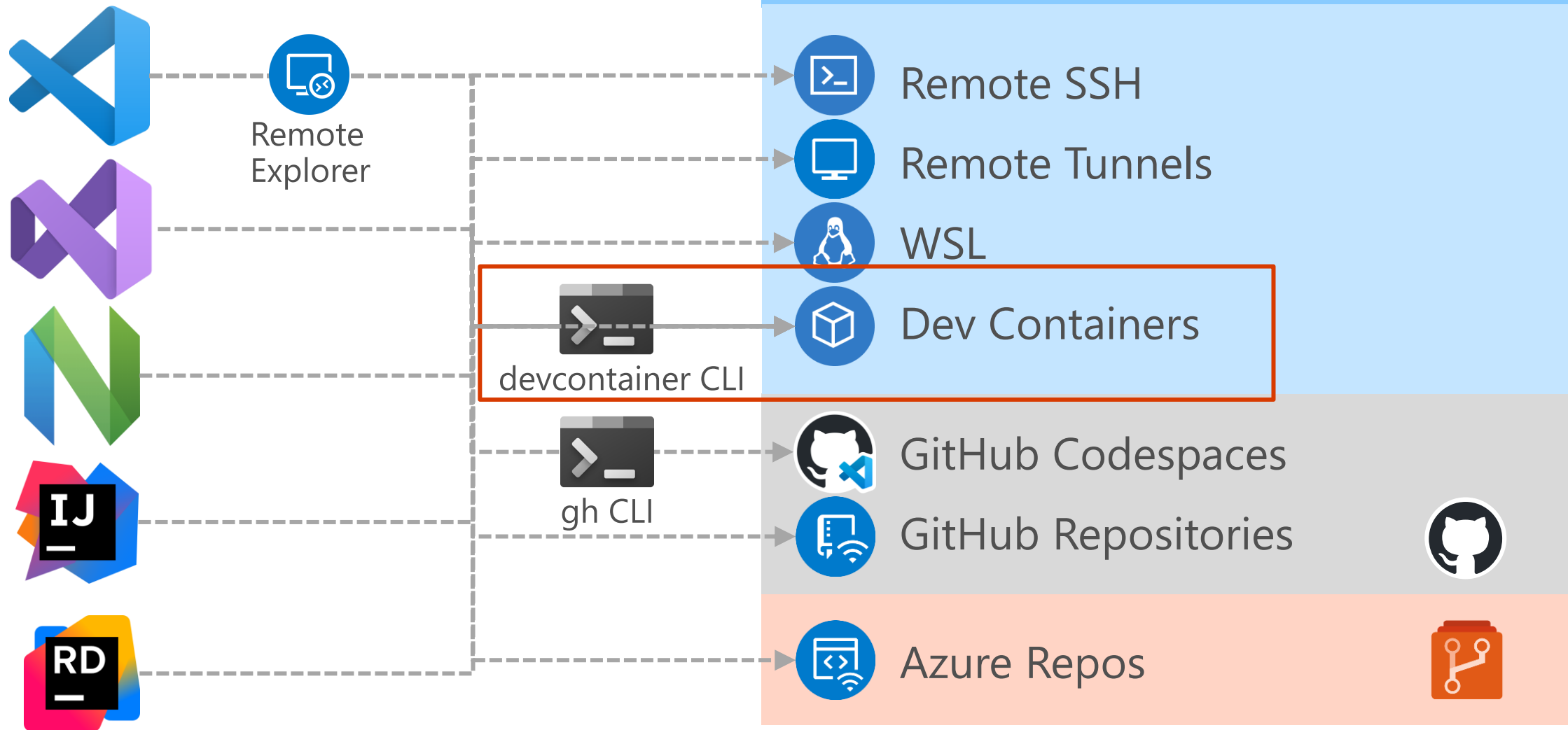
Running a container as
development environment



Fundamentals of Dev Containers



Connecting remotely



Demo

Remote Explorer



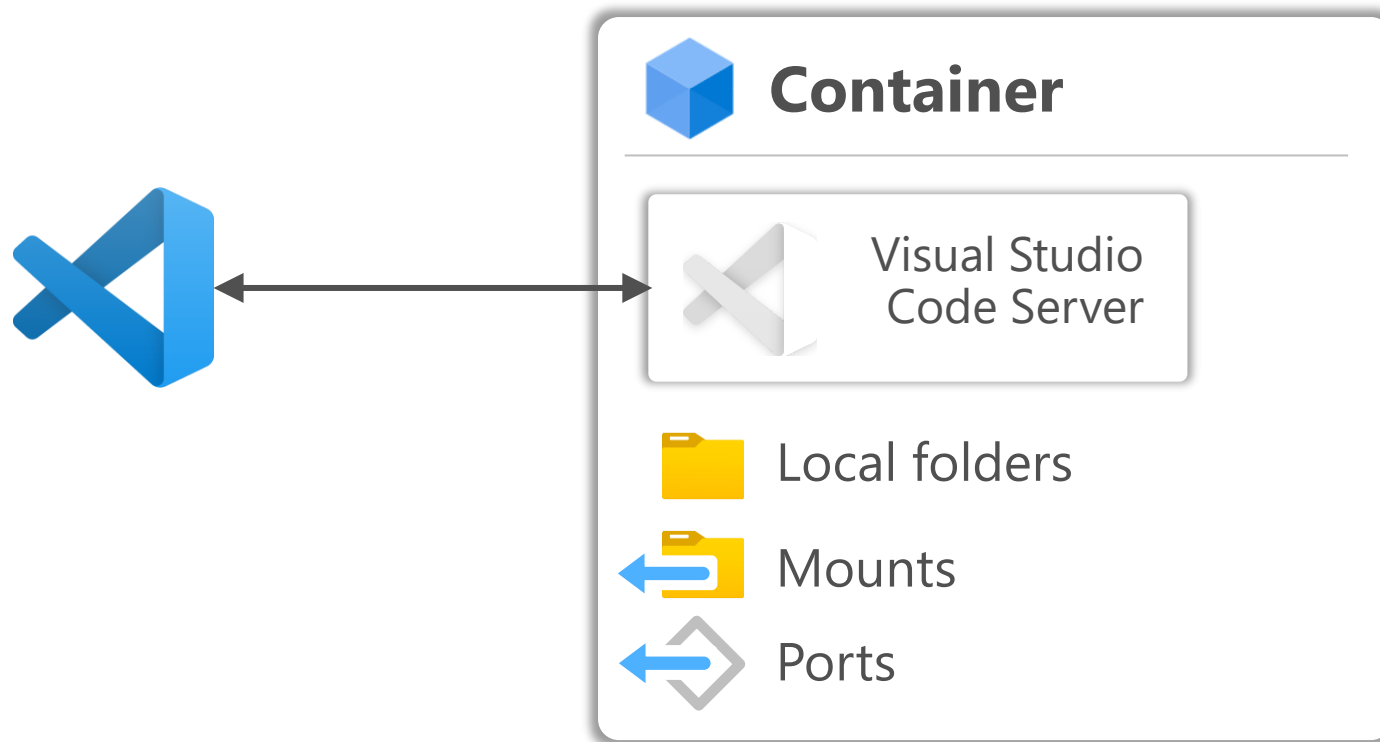
Connecting to running container



Connecting to any container

Prerequisites for Linux distro apply

Running on local or remote machine,
Kubernetes or in Docker Compose



Automatically installs

Visual Studio Code Server

 `/vscode/vscode-server`
`/root/.vscode-server`

Default extensions

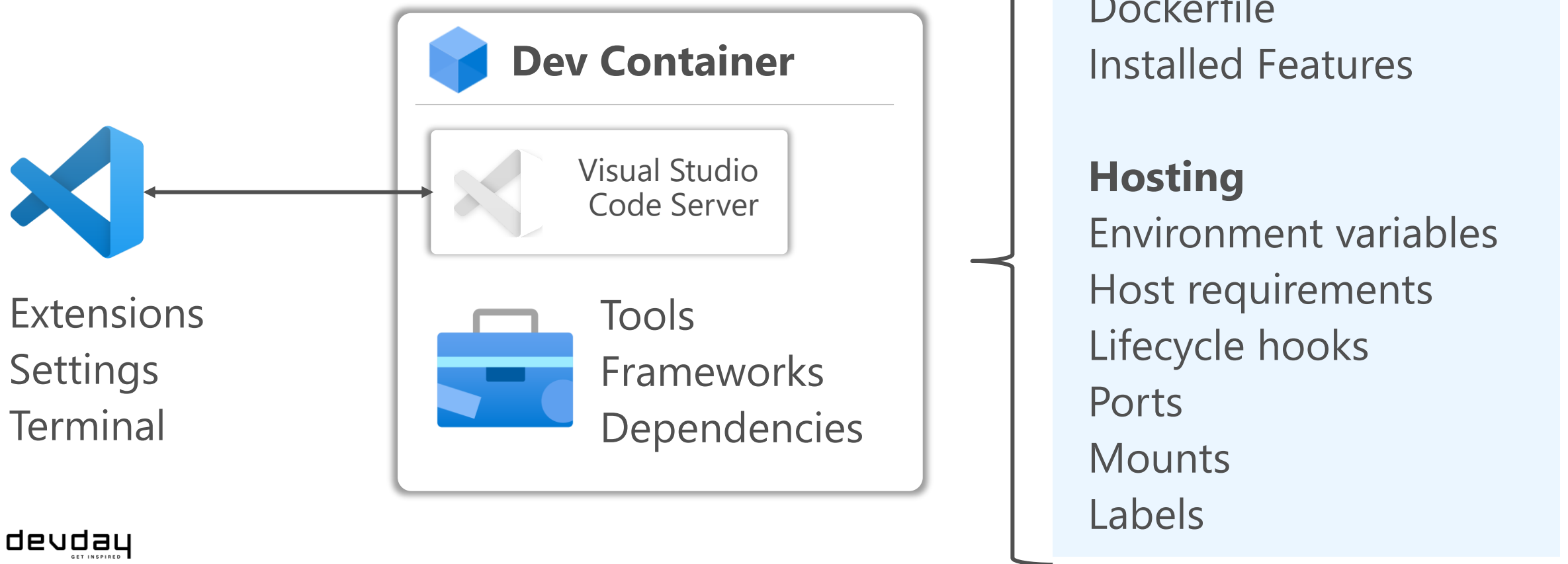
GitHub Copilot

GitHub Copilot Chat

GitHub Pull Requests

Configurable in
Settings

Anatomy of a Dev Container



Choosing your dev container strategy

1. Regular Container

- Manual creation
- Harder management
- Limited development experience

2. Base image

- Leverage existing dev container images
- Easy to get started in most situations

3. Custom Dockerfile

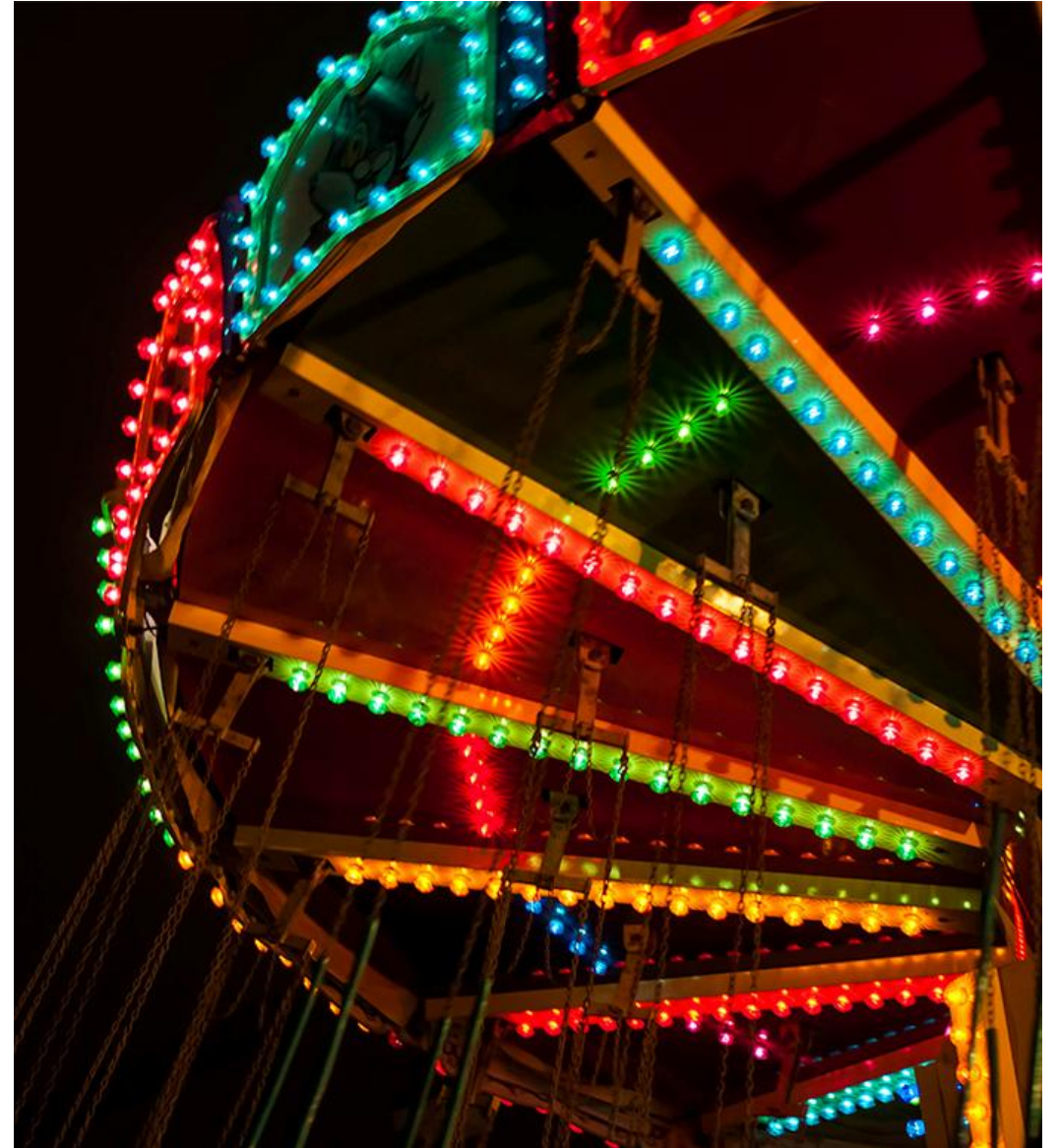
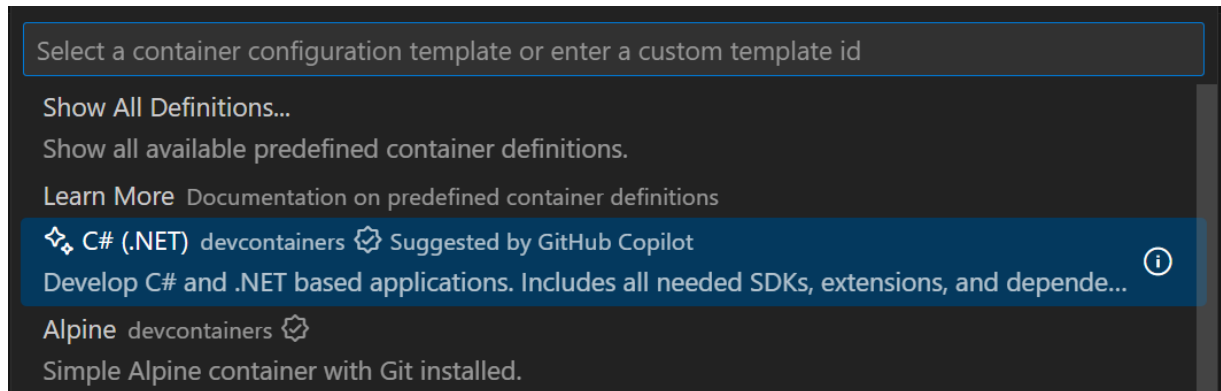
- Create new image base from scratch
- Enhance dev container images

4. Docker Compose

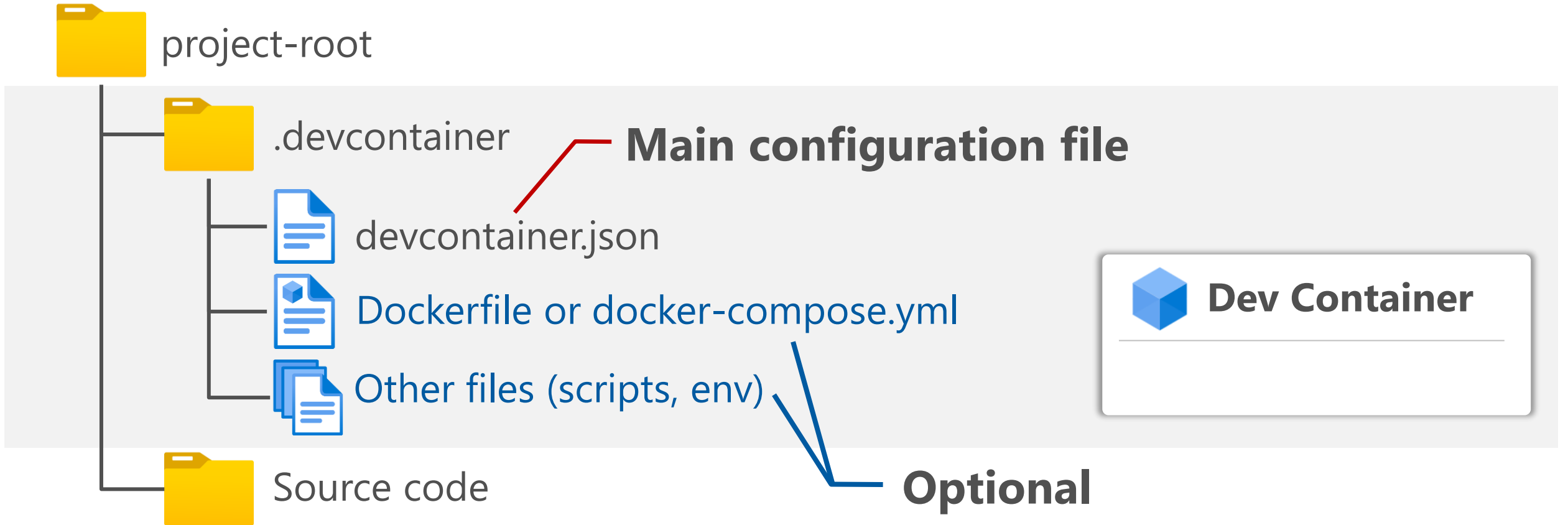
- Use compositions for complex projects
- Run dependencies from container
- For mono-repo in heterogeneous technologies

Demo

Defining Dev Containers



Defining your Dev Container as code



Location of container configuration files



Workspace

Resides within root of source code folder
Most common scenario



User data

Centralized location outside of source code
Keeps configuration separate



Image

Configuration based on container image name
Automatically applied when attaching



Repository

Matches origin of remote repository
For repositories outside of your control



Find it at:

```
%APPDATA%\Roaming\Code\User\  
globalStorage\  
ms-vscode-remote.remote-containers
```

Folders:

configs
imageConfigs
repositoryConfigs (create yourself)

Development container specification

“provide a way to enrich containers with the content and metadata necessary to enable development inside them”

Originally by  **Microsoft** and Visual Studio Code team

Now community driven

Define repeatable development environments

Metadata to drive tooling and dev experiences


JSON with comments file (**devcontainer.json**)

Image labels used as an array of metadata snippets


Specification

Dev Container metadata reference

The `devcontainer.json` file contains any needed metadata and settings required to configure a **development container** for a given well-defined tool and runtime stack. It can be used by [tools and services that support the dev container spec](#) to create a **development environment** that contains one or more **development containers**.

Metadata properties marked with a  can be stored in the `devcontainer.metadata` **container image label** in addition to `devcontainer.json`. This label can contain an array of json snippets that will be automatically merged with `devcontainer.json` contents (if any) when a container is created.

General devcontainer.json properties

Property	Type	Description
<code>name</code>	string	A name for the dev container displayed in the UI.
<code>forwardPorts</code> 	array	An array of port numbers or <code>"host:port"</code> values (e.g. <code>[3000, "db:5432"]</code>) that should always be forwarded from inside the primary container to the local

Microsoft base images for Dev Containers

mcr.microsoft.com/devcontainers/base



Alpine: 3.17-3.20,

Debian: 9 (stretch), 10 (buster), 11 (bullseye), 12 (bookworm)

Ubuntu: 18 (bionic), 20 (focal), 22 (jammy), 24 (noble)

Semantic versioning for unique combinations:

OS version

Toolset (differs per OS)

Example: **base**:1.1.0-ubuntu-24.04 or just **base**:ubuntu



Zsh, Oh my zsh, non-root user **vscode**



Common dev tools and dependencies

Configurable
root or base image user

Multiple devcontainer definitions

Organize in subfolders

Searches in several locations

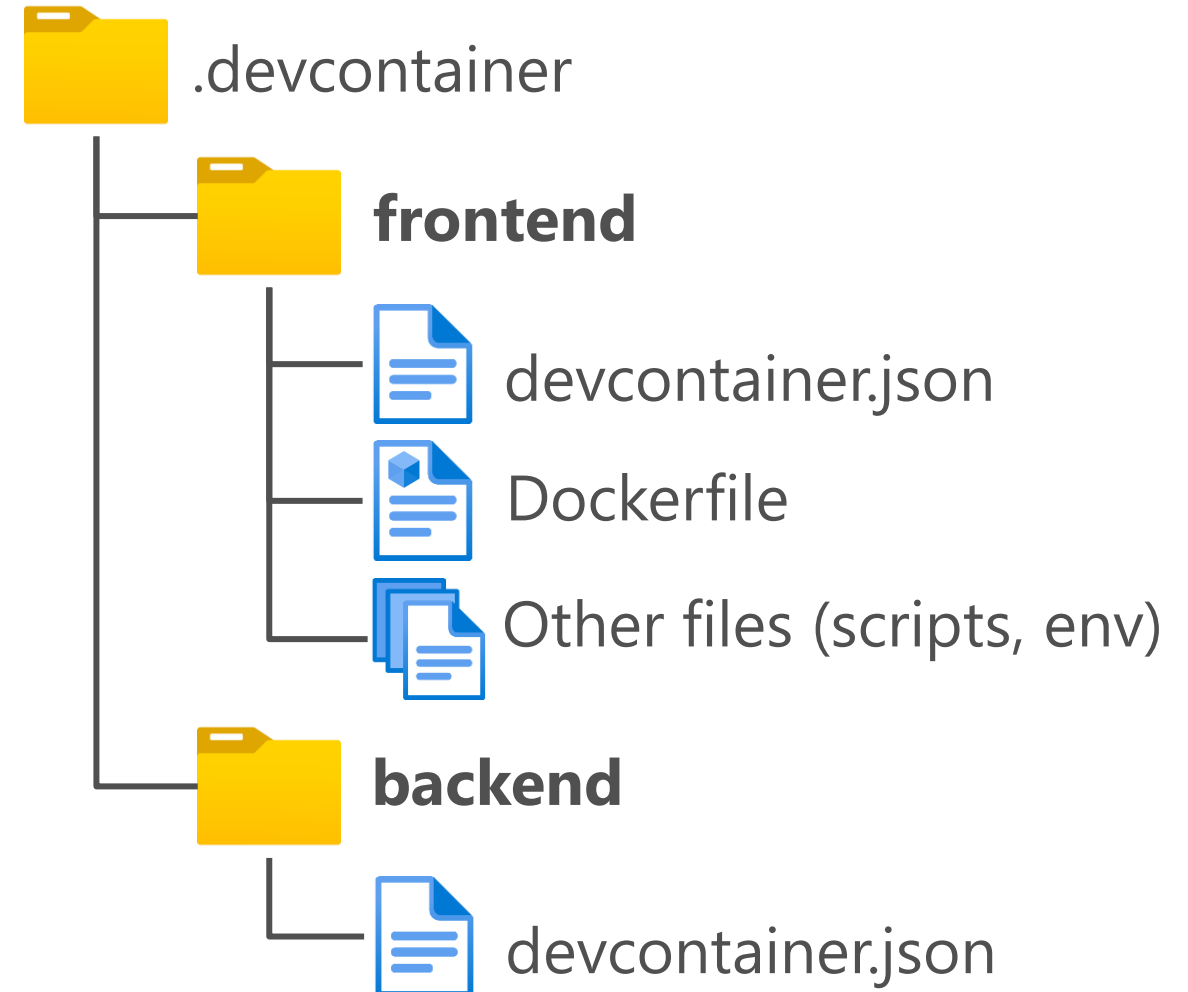
 **.devcontainer** folder
and subfolders

 **.devcontainer.json** in root

 **With a dot (not a typo)**

Ability to switch easily

>Dev Containers: Switch container



Templates for Dev Containers



Distributions Alpine, Ubuntu, Debian



Universal

Many frameworks and tools (.NET, Node, Python, PHP, Ruby, Java, ...)

Default template for GitHub Codespaces



Language and framework starting points

Docker enabled (Docker or Kubernetes)

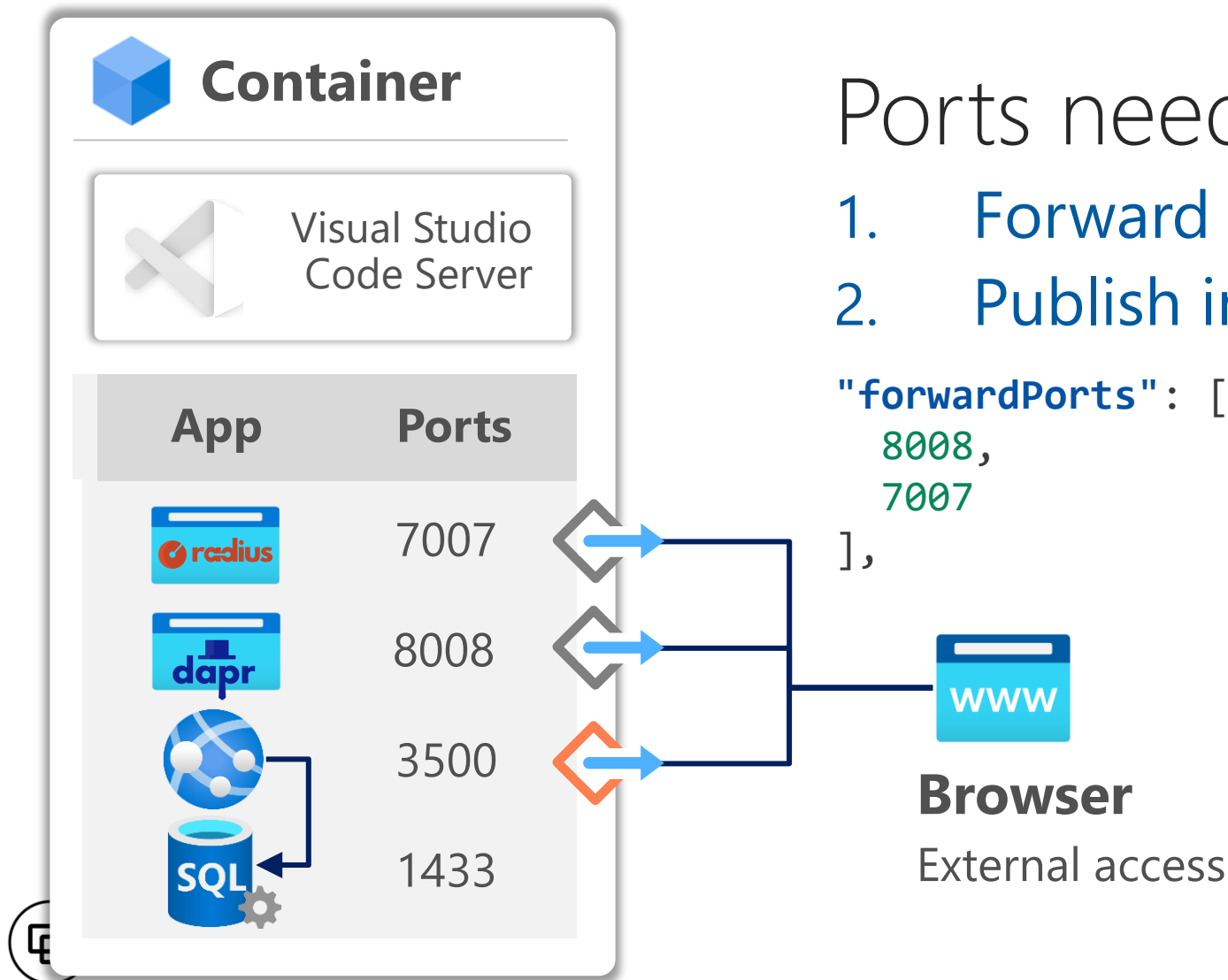
C#, C++, F#, Java, PHP, Python, Ruby, Rust, Node, Jekyll, Markdown



Community templates

<https://containers.dev/templates>

Container ports



Ports need to be exposed

1. Forward manually or automatic
2. Publish in dev container definition

```
"forwardPorts": [  
  8008,  
  7007  
],
```

```
"portsAttributes": {  
  "7007": {  
    "label": "radius-dashboard",  
    "onAutoForward": "notify",  
    "protocol": "http"  
  },  
  1  
}
```

	Port	Forwarded Address	Running P
○	1337	localhost:1337	
●	backend (3000)	127.0.0.1:3000	rad run ./d
●	3500	127.0.0.1:3500	rad run ./d
●	radius-dashboard (70...	localhost:7007	rad run ./d
○	dapr-dashboard (8008)	localhost:8008	

Demo

Developing for Radius




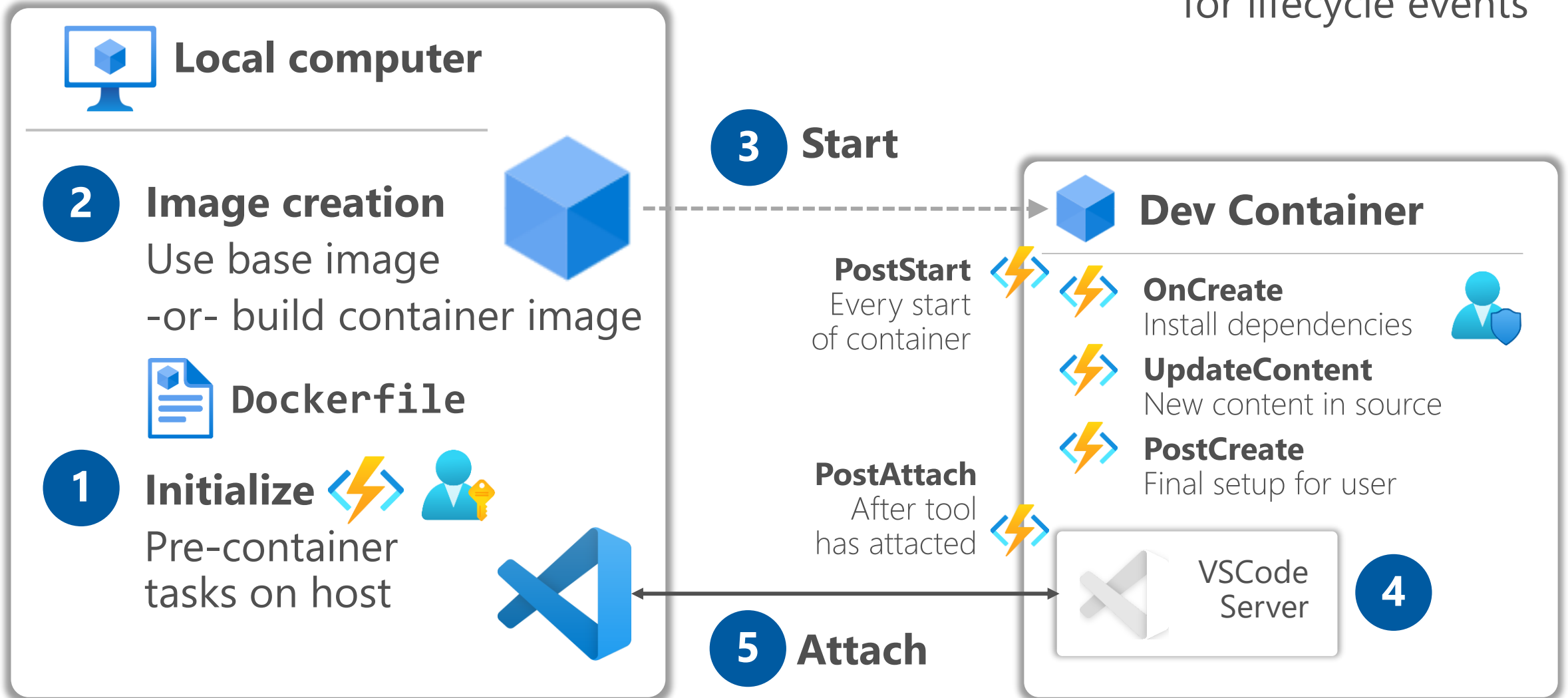
Install: Kubernetes cluster, dapr, radius

Extra CLIs: kubectl, helm, k3d

Visual Studio Code: Extensions and settings

Lifecycle of a Dev Container

 **devcontainer.json**
Defines commands
for lifecycle events



Templates and **features**



Dev Container templates and features

Apply templates and features to your repository



```
devcontainer templates apply  
--template-id ghcr.io/alexthissen/devcontainers/radius:1.1
```



Pre-packaged set of files

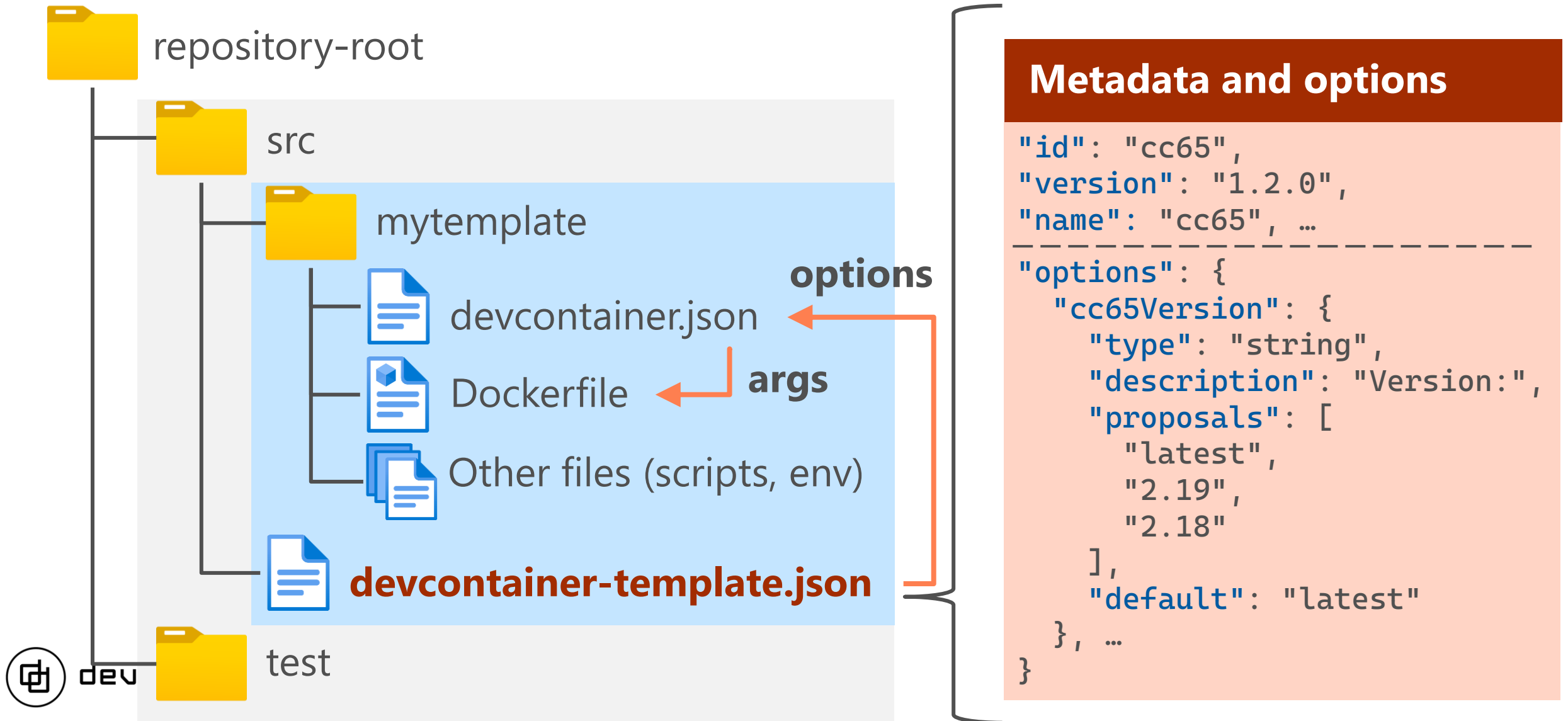
Published tarballs

Different structure for templates and features

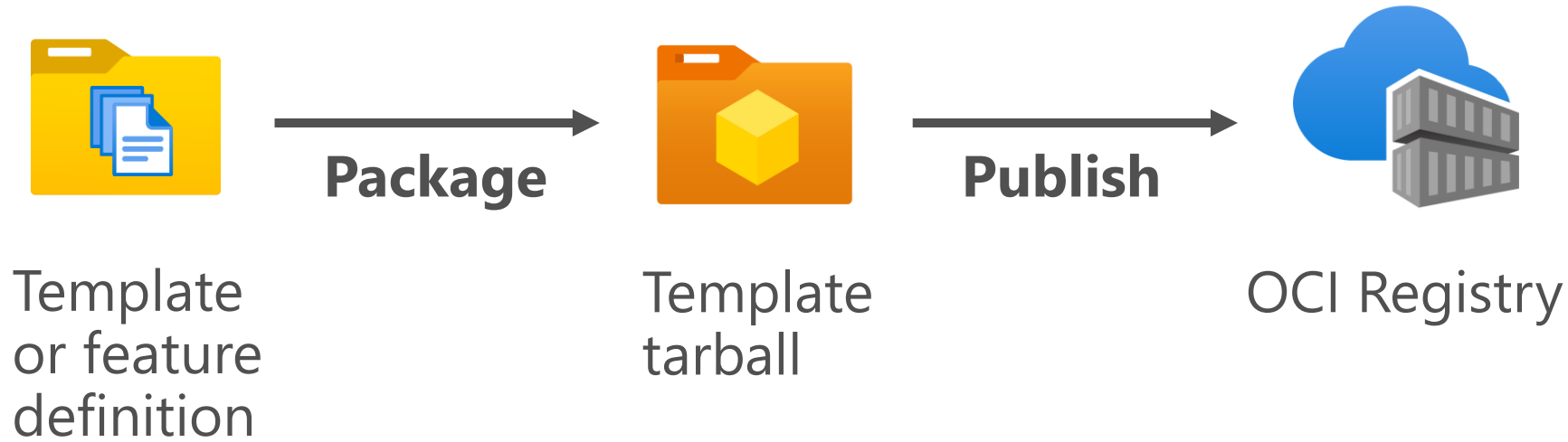
Use existing templates and features

Build your own

Defining your template for a Dev Container



Creating devcontainer templates and features



Inception

Use a devcontainer to create a template or features

Template and feature starter repositories

<https://github.com/devcontainers/template-starter> and [feature-starter](https://github.com/devcontainers/feature-starter)

DevContainer features

What is a feature?

Runnable scripts during image creation after Dockerfile

Many features available

Docker in Docker, Docker outside Docker

Tools: Kubectl, Helm, Minikube

CLI: Dapr, Azure, GitHub, AWS, .NET

Languages: PHP, Go, Ruby, Rust, C++, ...

Build your own feature

Prebuild packages (like templates)

Local features

```
"features": {  
  "ghcr.io/.../docker-in-docker:2": {},  
  "ghcr.io/.../azure-cli:1": {},  
  "ghcr.io/.../github-cli:1": {},  
  "ghcr.io/rio/features/k3d:1": {}  
},
```

```
"features": {  
  "./local-features/git": { }  
}
```

Example feature: Common-utils

Installed for each base dev container image

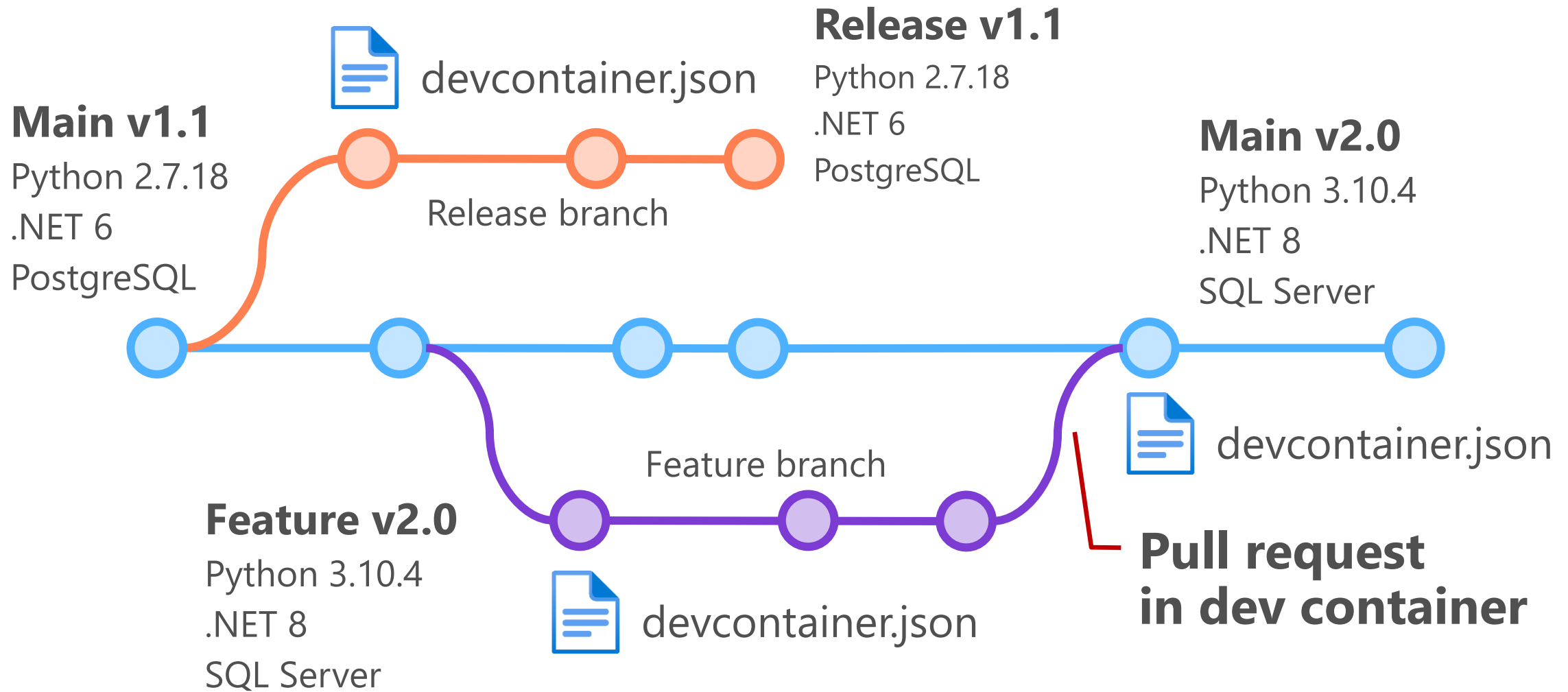
Alpine, Debian, Ubuntu

```
"features": {  
  "ghcr.io/devcontainers/features/common-utils:2": {  
    "installZsh": "true",  
    "configureZshAsDefaultShell": "false",  
    "username": "vscode",  
    "userUid": "1000",  
    "userGid": "1000",  
    "upgradePackages": "true"  
  }  
}
```

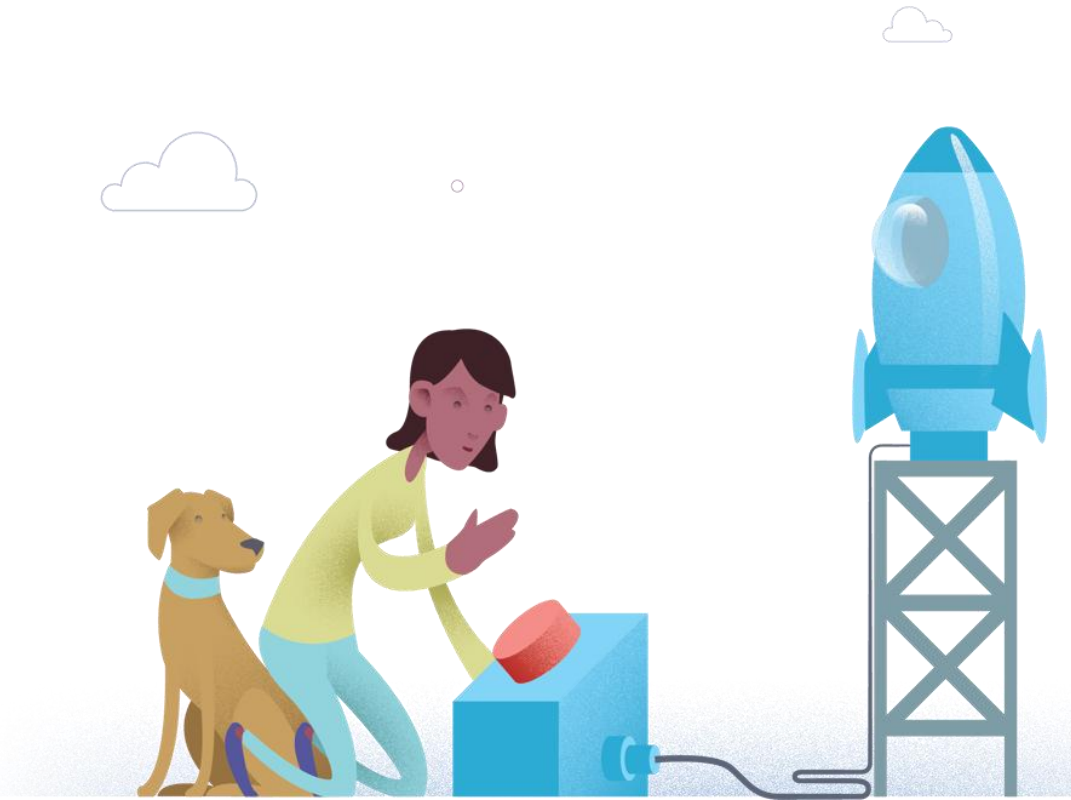

Working with **Code** and **Automation** for Dev Containers



Branching



Prebuilding



Benefits to pre-build
your dev container images:

Faster **startup** times

Simplifies configuration

Specific **version** of tools

Improve **supply-chain security** and
avoid potential breaks

Automation

Leverage GitHub Actions and Azure Pipelines



Action: **devcontainer/actions** and **devcontainer/ci**



Task: **DevcontainersCi**

Publish

templates and features

Prebuild

dev container images

Build

from dev container



- name: **Run make ci-build in dev container**
uses: **devcontainers/ci@v0.3**
with:
 cacheFrom: **ghcr.io/alexthissen/devcontainer/radius**
 push: **never**
 runCmd: **make ci-build**

Building in devcontainer from pipeline

Workflow permissions

Choose the default permissions granted to the GITHUB_TOKEN when running workflows in this organization. You can specify more granular permissions in the workflow using YAML. [Learn more about managing permissions.](#)

Repository administrators will only be able to change the default permissions to a more restrictive setting.

☒ **Read and write permissions**

Workflows have read and write permissions in the repository for all scopes.

☐ **Read repository contents and packages permissions**

Workflows have read permissions in the repository for the contents and packages scopes only.

Choose whether GitHub Actions can create pull requests or submit approving pull request reviews.

☒ **Allow GitHub Actions to create and approve pull requests**

Save



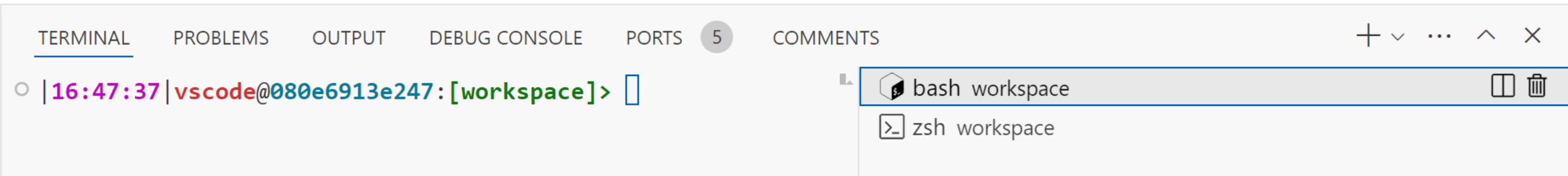
Required for pipelines to write packages

<https://github.com/organizations/AtariLynx/settings/actions>

Personalization of your development environment



Dotfiles ~/.*



Many files to customize your environment

bash, ash, zsh and fish terminal look and feel

Git behavior

All start with a dot (.)

Use **ls -la** to see them

Common files

- .bash_profile
- .bashrc
- .profile
- .zshenv
- .zprofile
- .zshrc
- .zlogin
- .gitconfig

Dotfiles support

dotfiles repository can be cloned into devcontainer

```
"dotfiles.repository": "https://github.com/alexthissen/dotfiles.git",  
"dotfiles.installCommand": "",  
"dotfiles.targetPath": "~/dotfiles",
```

1. Search for installation shell files
2. If none present, symlink all dot files to ~

Installation files



- install.sh
- install
- bootstrap.sh
- bootstrap
- script/bootstrap
- setup.sh
- setup
- script/setup

Separate files for bash and zsh

Configuration and settings

1. Customizations per tool

```
"customizations": {  
  "vscode": {  
    "settings": {  
      "terminal.integrated.shell.linux": "/bin/ash",  
      "hexeditor.columnWidth": 32,  
      "hexeditor.defaultEndianness": "little",  
      "makefile.configureAfterCommand": false  
    },  
    "extensions": [  
      "ms-vscode.makefile-tools",  
      "ms-vscode.hexeditor"  
    ]  
  }  
},
```

Extension settings
Configuration per
extension

Extensions
Specific to dev container

2. Settings sync

Oh-my-zsh

Automatically installed by common-utils

Included in Microsoft base images

Customize files .zshrc and .zprofile

Choose theme `ZSH_THEME="agnoster"`

For example: **Agnoster** (<https://github.com/agnoster/agnoster-zsh-theme>)

Reference



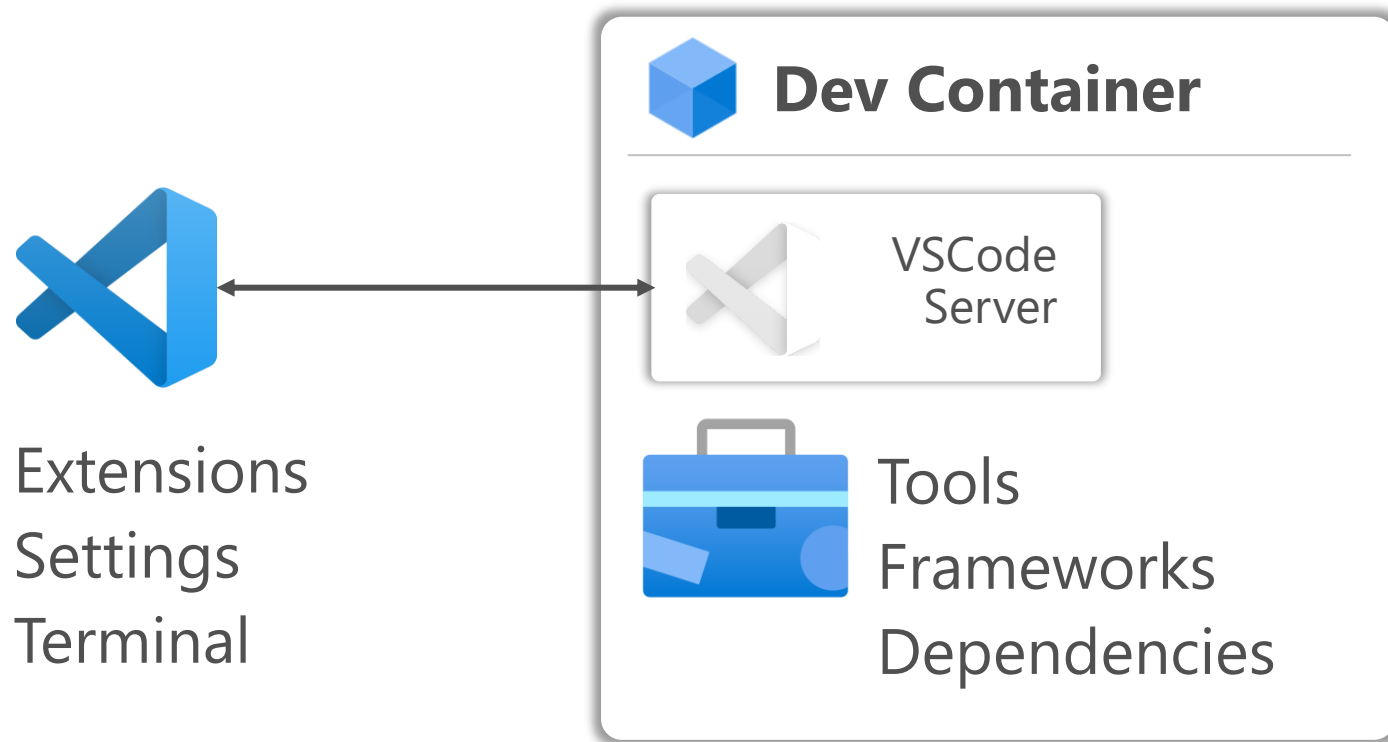
<https://github.com/devcontainers/>

<https://containers.dev/>

<https://code.visualstudio.com/docs/devcontainers/containers>

<https://code.visualstudio.com/remote/advancedcontainers/overview>

Summary



Thanks for attending!

Alex Thissen

@alexthissen

Cloud architect, Xebia Microsoft Services



<https://openfeedback.io/dd2024/2024-11-07/686614>