Reusing Dev Containers Within a Pipeline

Given a repository with a local development container a.k.a. dev container that contains all the tooling required for development, would it make sense to reuse that container for running the tooling in the Continuous Integration pipelines?

Options for Building Dev Containers Within a Pipeline

There are three ways to build devcontainers within pipeline:

- With GitHub - devcontainers/ci builds the container with the `devcontainer.json`. Example here: devcontainers/ci · Getting Started.
- With GitHub - devcontainers/cli, which is the same as the above, but using the underlying CLI directly without tasks.
- Building the `DockerFile` with `docker build`. This option excludes all configuration/features specified within the `devcontainer.json`.

Considered Options

- Run CI pipelines in the native environment
- Run CI pipelines in the dev container via building image locally
- Run CI pipelines in the dev container with a container registry

Here are below pros and cons for both approaches:

Run CI Pipelines in the Native Environment

Pros	Cons
Can use any pipeline tasks available	Need to keep two sets of tooling and their versions in sync

Pros	Cons
No container registry	Can take some time to start, based on tools/dependencies required
Agent will always be up to date with security patches	The dev container should always be built within each run of the CI pipeline, to verify the changes within the branch haven't broken anything

Run CI Pipelines in the Dev Container Without Image Caching

Pros	Cons
Utilities scripts will work out of the box	Need to rebuild the container for each run, given that there may be changes within the branch being built
Rules used (for linting or unit tests) will be the same on the CI	Not everything in the container is needed for the CI pipeline[1]
No surprise for the developers, local outputs (of linting for instance) will be the same in the CI	Some pipeline tasks will not be available
All tooling and their versions defined in a single place	Building the image for each pipeline run is slow[2]
Tools/dependencies are already present	
The dev container is being tested to include all new tooling in addition to not being broken	

> [1]: container size can be reduced by exporting the layer that contains only the tooling needed for the CI pipeline
>
> [2]: could be mitigated via adding image caching without using a container registry

Run CI Pipelines in the Dev Container with Image Registry

Pros	Cons
Utilities scripts will work out of the box	Need to rebuild the container for each run, given that there may be changes within the branch being built
No surprise for the developers, local outputs (of linting for instance) will be the same in the CI	Not everything in the container is needed for the CI pipeline[1]
Rules used (for linting or unit tests) will be the same on the CI	Some pipeline tasks will not be available[2]
All tooling and their versions defined in a single place	Require access to a container registry to host the container within the pipeline[3]
Tools/dependencies are already present	
The dev container is being tested to include all new tooling in addition to not being broken	
Publishing the container built from `devcontainer.json` allows you to reference it in the cacheFrom in `devcontainer.json` (see docs). By doing this, VS Code will use the published image as a layer cache when building	

> [1]: container size can be reduces by exporting the layer that contains only the tooling needed for the CI pipeline. This would require building the image without tasks
>
> [2]: using container jobs in AzDO you can use all tasks (as far as I can tell). Reference: Dockerizing DevOps V2 - AzDO container jobs - DEV Community
>
> [3]: within GH actions, the default Github Actions token can be used for accessing GHCR without setting up separate registry, see the example below. **Note:** This does not build the `Dockerfile` together with the `devcontainer.json`

```yaml
    - uses: whoan/docker-build-with-cache-action@v5
      id: cache
      with:
        username: $GITHUB_ACTOR
        password: "${{ secrets.GITHUB_TOKEN }}"
        registry: docker.pkg.github.com
        image_name: devcontainer
        dockerfile: .devcontainer/Dockerfile
```

Last update: August 22, 2024