Custom Dockerfile syntax

Dockerfile frontend

BuildKit supports loading frontends dynamically from container images. To use an external Dockerfile frontend, the first line of your Dockerfile needs to set the `syntax` directive pointing to the specific image you want to use:

```
# syntax=[remote image reference]
```

For example:

```
# syntax=docker/dockerfile:1
# syntax=docker.io/docker/dockerfile:1
# syntax=example.com/user/repo:tag@sha256:abcdef...
```

You can also use the predefined `BUILDKIT_SYNTAX` build argument to set the frontend image reference on the command line:

```
$ docker build --build-arg BUILDKIT_SYNTAX=docker/dockerfile:1 .
```

This defines the location of the Dockerfile syntax that is used to build the Dockerfile. The BuildKit backend allows seamlessly using external implementations that are distributed as Docker images and execute inside a container sandbox environment.

Custom Dockerfile implementations allow you to:

- Automatically get bug fixes without updating the Docker daemon
- Make sure all users are using the same implementation to build your Dockerfile
- Use the latest features without updating the Docker daemon
- Try out new features or third-party features before they are integrated in the Docker daemon
- Use alternative build definitions, or create your own
- Build your own Dockerfile frontend with custom features

> **Note**
>
> BuildKit ships with a built-in Dockerfile frontend, but it's recommended to use an external image to make sure that all users use the same version on the builder and to pick up bug fixes automatically without waiting for a new version of BuildKit or Docker Engine.

Official releases

Docker distributes official versions of the images that can be used for building Dockerfiles under `docker/dockerfile` repository on Docker Hub. There are two channels where new images are released: `stable` and `labs`.

Stable channel

The `stable` channel follows semantic versioning. For example:

- `docker/dockerfile:1` - kept updated with the latest `1.x.x` minor *and* patch release.
- `docker/dockerfile:1.2` - kept updated with the latest `1.2.x` patch release, and stops receiving updates once version `1.3.0` is released.
- `docker/dockerfile:1.2.1` - immutable: never updated.

We recommend using `docker/dockerfile:1`, which always points to the latest stable release of the version 1 syntax, and receives both "minor" and "patch" updates for the version 1 release cycle. BuildKit automatically checks for updates of the syntax when performing a build, making sure you are using the most current version.

If a specific version is used, such as `1.2` or `1.2.1`, the Dockerfile needs to be updated manually to continue receiving bugfixes and new features. Old versions of the Dockerfile remain compatible with the new versions of the builder.

Labs channel

The `labs` channel provides early access to Dockerfile features that are not yet available in the `stable` channel. `labs` images are released at the same time as stable releases, and follow the same version pattern, but use the `-labs` suffix, for example:

- `docker/dockerfile:labs` - latest release on `labs` channel.
- `docker/dockerfile:1-labs` - same as `dockerfile:1`, with experimental features enabled.
- `docker/dockerfile:1.2-labs` - same as `dockerfile:1.2`, with experimental features enabled.
- `docker/dockerfile:1.2.1-labs` - immutable: never updated. Same as `dockerfile:1.2.1`, with experimental features enabled.

Choose a channel that best fits your needs. If you want to benefit from new features, use the `labs` channel. Images in the `labs` channel contain all the features in the `stable` channel, plus early access features. Stable features in the `labs` channel follow semantic versioning, but early access features don't, and newer releases may not be backwards compatible. Pin the version to avoid having to deal with breaking changes.

Other resources

For documentation on `labs` features, master builds, and nightly feature releases, refer to the description in the BuildKit source repository on GitHub. For a full list of available images, visit the `docker/dockerfile` repository on Docker Hub, and the `docker/dockerfile-upstream` repository on Docker Hub for development builds.