Configure BuildKit

- Registry mirror
- Setting registry certificates
- CNI networking
  - Bridge networking
  - Custom CNI configuration
- Resource limiting
  - Max parallelism
  - TCP connection limit

If you create a `docker-container` or `kubernetes` builder with Buildx, you can apply a custom BuildKit configuration by passing the `--buildkitd-config` flag to the `docker buildx create` command.

Registry mirror

You can define a registry mirror to use for your builds. Doing so redirects BuildKit to pull images from a different hostname. The following steps exemplify defining a mirror for `docker.io` (Docker Hub) to `mirror.gcr.io`.

1. Create a TOML at `/etc/buildkitd.toml` with the following content:

   ```
   debug = true
   [registry."docker.io"]
     mirrors = ["mirror.gcr.io"]
   ```

   > [!NOTE]
   > `debug = true` turns on debug requests in the BuildKit daemon, which logs a message that shows when a mirror is being used.

2. Create a `docker-container` builder that uses this BuildKit configuration:

```
$ docker buildx create --use --bootstrap \
  --name mybuilder \
  --driver docker-container \
  --buildkitd-config /etc/buildkitd.toml
```

3. Build an image:

```
docker buildx build --load . -f - <<EOF
FROM alpine
RUN echo "hello world"
EOF
```

The BuildKit logs for this builder now shows that it uses the GCR mirror. You can tell by the fact that the response messages include the `x-goog-*` HTTP headers.

```
$ docker logs buildx_buildkit_mybuilder0
```

```
...
time="2022-02-06T17:47:48Z" level=debug msg="do request"
request.header.accept="application/vnd.docker.container.image.v1+json, */*"
request.header.user-agent=containerd/1.5.8+unknown request.method=GET spanID=9460e5b6e64cec91
traceID=b162d3040ddf86d6614e79c66a01a577
time="2022-02-06T17:47:48Z" level=debug msg="fetch response received" response.header.accept-
ranges=bytes response.header.age=1356 response.header.alt-svc="h3=\":443\"; ma=2592000,h3-
29=\":443\"; ma=2592000,h3-Q050=\":443\"; ma=2592000,h3-Q046=\":443\"; ma=2592000,h3-
Q043=\":443\"; ma=2592000,quic=\":443\"; ma=2592000; v=\"46,43\"" response.header.cache-
control="public, max-age=3600" response.header.content-length=1469 response.header.content-
type=application/octet-stream response.header.date="Sun, 06 Feb 2022 17:25:17 GMT"
response.header.etag="\"774380abda8f4eae9a149e5d5d3efc83\"" response.header.expires="Sun, 06
Feb 2022 18:25:17 GMT" response.header.last-modified="Wed, 24 Nov 2021 21:07:57 GMT"
response.header.server=UploadServer response.header.x-goog-generation=1637788077652182
response.header.x-goog-hash="crc32c=V3DSrg==" response.header.x-goog-
hash.1="md5=d0OAq9qPTq6aFJ5dXT78gw==" response.header.x-goog-metageneration=1
response.header.x-goog-storage-class=STANDARD response.header.x-goog-stored-content-
encoding=identity response.header.x-goog-stored-content-length=1469 response.header.x-
guploader-
uploadid=ADPycduqQipVAXc3tzXmTzKQ2gTT6CV736B2J628smtD1iDytEyiYCgvvdD8zz9BT1J1sASUq9pW_ctUyC4B-
v2jvhIxnZTlKg response.status="200 OK" spanID=9460e5b6e64cec91
traceID=b162d3040ddf86d6614e79c66a01a577
```

time="2022-02-06T17:47:48Z" level=debug msg="fetch response received" response.header.accept-ranges=bytes response.header.age=760 response.header.alt-svc="h3=\":443\"; ma=2592000,h3-29=\":443\"; ma=2592000,h3-Q050=\":443\"; ma=2592000,h3-Q046=\":443\"; ma=2592000,h3-Q043=\":443\"; ma=2592000,quic=\":443\"; ma=2592000; v=\"46,43\"" response.header.cache-control="public, max-age=3600" response.header.content-length=1471 response.header.content-type=application/octet-stream response.header.date="Sun, 06 Feb 2022 17:35:13 GMT" response.header.etag="\"35d688bd15327daafcdb4d4395e616a8\"" response.header.expires="Sun, 06 Feb 2022 18:35:13 GMT" response.header.last-modified="Wed, 24 Nov 2021 21:07:12 GMT" response.header.server=UploadServer response.header.x-goog-generation=1637788032100793 response.header.x-goog-hash="crc32c=aWgRjA==" response.header.x-goog-hash.1="md5=NdaIvRUyfar8201DleYWqA==" response.header.x-goog-metageneration=1 response.header.x-goog-storage-class=STANDARD response.header.x-goog-stored-content-encoding=identity response.header.x-goog-stored-content-length=1471 response.header.x-guploader-uploadid=ADPycdtR-gJYwC7yHquIkJWFFG8FovDySvtmRnZBqlO3yVDanBXh_VqKYt400yhuf0XbQ3ZMB9IZV2vlcyHezn_Pu3a1SMMtiw response.status="200 OK" spanID=9460e5b6e64cec91 traceID=b162d3040ddf86d6614e79c66a01a577
time="2022-02-06T17:47:48Z" level=debug msg=fetch spanID=9460e5b6e64cec91 traceID=b162d3040ddf86d6614e79c66a01a577
time="2022-02-06T17:47:48Z" level=debug msg=fetch spanID=9460e5b6e64cec91 traceID=b162d3040ddf86d6614e79c66a01a577
time="2022-02-06T17:47:48Z" level=debug msg=fetch spanID=9460e5b6e64cec91 traceID=b162d3040ddf86d6614e79c66a01a577
time="2022-02-06T17:47:48Z" level=debug msg=fetch spanID=9460e5b6e64cec91 traceID=b162d3040ddf86d6614e79c66a01a577
time="2022-02-06T17:47:48Z" level=debug msg="do request" request.header.accept="application/vnd.docker.image.rootfs.diff.tar.gzip, */*" request.header.user-agent=containerd/1.5.8+unknown request.method=GET spanID=9460e5b6e64cec91 traceID=b162d3040ddf86d6614e79c66a01a577
time="2022-02-06T17:47:48Z" level=debug msg="fetch response received" response.header.accept-ranges=bytes response.header.age=1356 response.header.alt-svc="h3=\":443\"; ma=2592000,h3-29=\":443\"; ma=2592000,h3-Q050=\":443\"; ma=2592000,h3-Q046=\":443\"; ma=2592000,h3-Q043=\":443\"; ma=2592000,quic=\":443\"; ma=2592000; v=\"46,43\"" response.header.cache-control="public, max-age=3600" response.header.content-length=2818413 response.header.content-type=application/octet-stream response.header.date="Sun, 06 Feb 2022 17:25:17 GMT" response.header.etag="\"1d55e7be5a77c4a908ad11bc33ebea1c\"" response.header.expires="Sun, 06 Feb 2022 18:25:17 GMT" response.header.last-modified="Wed, 24 Nov 2021 21:07:06 GMT" response.header.server=UploadServer response.header.x-goog-generation=1637788026431708 response.header.x-goog-hash="crc32c=ZojF+g==" response.header.x-goog-hash.1="md5=HVXnvlp3xKkIrRG8M+vqHA==" response.header.x-goog-metageneration=1

```
response.header.x-goog-storage-class=STANDARD response.header.x-goog-stored-content-
encoding=identity response.header.x-goog-stored-content-length=2818413 response.header.x-
guploader-uploadid=ADPycdsebqxiTBJqZ0bv9zBigjFxgQydD2ESZSkKchpE0ILlN9Ibko3C5r4fJTJ4UR9ddp-UBd-
2v_4eRpZ8Yo2llW_j4k8WhQ response.status="200 OK" spanID=9460e5b6e64cec91
traceID=b162d3040ddf86d6614e79c66a01a577
...
```

Setting registry certificates

If you specify registry certificates in the BuildKit configuration, the daemon copies the files into the container under `/etc/buildkit/certs`. The following steps show adding a self-signed registry certificate to the BuildKit configuration.

1. Add the following configuration to `/etc/buildkitd.toml`:

   ```
   # /etc/buildkitd.toml
   debug = true
   [registry."myregistry.com"]
     ca=["/etc/certs/myregistry.pem"]
     [[registry."myregistry.com".keypair]]
       key="/etc/certs/myregistry_key.pem"
       cert="/etc/certs/myregistry_cert.pem"
   ```

   This tells the builder to push images to the `myregistry.com` registry using the certificates in the specified location (`/etc/certs`).
2. Create a `docker-container` builder that uses this configuration:

   ```
   $ docker buildx create --use --bootstrap \
     --name mybuilder \
     --driver docker-container \
     --buildkitd-config /etc/buildkitd.toml
   ```

3. Inspect the builder's configuration file (`/etc/buildkit/buildkitd.toml`), it shows that the certificate configuration is now configured in the builder.

   ```
   $ docker exec -it buildx_buildkit_mybuilder0 cat /etc/buildkit/buildkitd.toml
   ```

   ```
   debug = true

   [registry]
   ```

```
  [registry."myregistry.com"]
    ca = ["/etc/buildkit/certs/myregistry.com/myregistry.pem"]

    [[registry."myregistry.com".keypair]]
      cert = "/etc/buildkit/certs/myregistry.com/myregistry_cert.pem"
      key = "/etc/buildkit/certs/myregistry.com/myregistry_key.pem"
```

4. Verify that the certificates are inside the container:

```
$ docker exec -it buildx_buildkit_mybuilder0 ls /etc/buildkit/certs/myregistry.com/
myregistry.pem    myregistry_cert.pem   myregistry_key.pem
```

Now you can push to the registry using this builder, and it will authenticate using the certificates:

```
$ docker buildx build --push --tag myregistry.com/myimage:latest .
```

CNI networking

CNI networking for builders can be useful for dealing with network port contention during concurrent builds.

Bridge networking

The BuildKit image ships with a built-in bridge network provider that uses a minimal set of bundled CNI plugins, so you don't need to build a custom image or supply your own CNI configuration. To use it, set the worker network mode to `bridge` when you create the builder:

```
$ docker buildx create --use --bootstrap \
  --name mybuilder \
  --driver docker-container \
  --buildkitd-flags "--oci-worker-net=bridge"
```

BuildKit creates a `buildkit0` bridge with a default subnet of `10.10.0.0/16`, and cleans up the bridge automatically when the daemon shuts down.

Custom CNI configuration

For more control over networking, build a custom BuildKit image with your own CNI configuration and plugins.

The following Dockerfile example shows a custom BuildKit image with CNI support. It uses the CNI config for integration tests in BuildKit as an example. Feel free to include your own CNI configuration.

```
# syntax=docker/dockerfile:1

ARG BUILDKIT_VERSION=v{{% param "buildkit_version" %}}
ARG CNI_VERSION=v1.0.1

FROM --platform=$BUILDPLATFORM alpine AS cni-plugins
RUN apk add --no-cache curl
ARG CNI_VERSION
ARG TARGETOS
ARG TARGETARCH
WORKDIR /opt/cni/bin
RUN curl -Ls
https://github.com/containernetworking/plugins/releases/download/$CNI_VERSION/cni-plugins-
$TARGETOS-$TARGETARCH-$CNI_VERSION.tgz | tar xzv

FROM moby/buildkit:${BUILDKIT_VERSION}
ARG BUILDKIT_VERSION
RUN apk add --no-cache iptables
COPY --from=cni-plugins /opt/cni/bin /opt/cni/bin
ADD https://raw.githubusercontent.com/moby/buildkit/${BUILDKIT_VERSION}/hack/fixtures/cni.json
/etc/buildkit/cni.json
```

Now you can build this image, and create a builder instance from it using the `--driver-opt image` option:

```
$ docker buildx build --tag buildkit-cni:local --load .
$ docker buildx create --use --bootstrap \
  --name mybuilder \
  --driver docker-container \
  --driver-opt "image=buildkit-cni:local" \
  --buildkitd-flags "--oci-worker-net=cni"
```

Resource limiting

Max parallelism

You can limit the parallelism of the BuildKit solver, which is particularly useful for low-powered machines, using a BuildKit configuration while creating a builder with the `--buildkitd-config` flag.

```
# /etc/buildkitd.toml
[worker.oci]
  max-parallelism = 4
```

Now you can create a `docker-container` builder that will use this BuildKit configuration to limit parallelism.

```
$ docker buildx create --use \
  --name mybuilder \
  --driver docker-container \
  --buildkitd-config /etc/buildkitd.toml
```

TCP connection limit

TCP connections are limited to 4 simultaneous connections per registry for pulling and pushing images, plus one additional connection dedicated to metadata requests. This connection limit prevents your build from getting stuck while pulling images. The dedicated metadata connection helps reduce the overall build time.

More information: moby/buildkit#2259

□□□ #2
□ □□□ □□□ 2026-10-31 19:04:27 UTC
□ □□□ □□□ 2026-10-31 19:04:29 UTC