Air-gapped containers

- Who should use air-gapped containers?
- How air-gapped containers work
- Prerequisites
- Configure air-gapped containers
  - Configuration parameters
  - Configuration examples
- Proxy Auto-Configuration (PAC) files
  - Basic PAC file structure
  - General considerations
  - PAC file return values
  - Advanced PAC file example
- Verify air-gapped container configuration
- Security considerations
- Next steps

{{< summary-bar feature_name="Air-gapped containers" >}}

Air-gapped containers let you restrict container network access by controlling where containers can send and receive data. This feature applies custom proxy rules to container network traffic, helping secure environments where containers shouldn't have unrestricted internet access.

Docker Desktop can configure container network traffic to accept connections, reject connections, or tunnel through HTTP or SOCKS proxies. You control which TCP ports the policy applies to and whether to use a single proxy or per-destination policies via Proxy Auto-Configuration (PAC) files.

Who should use air-gapped containers?

Use air-gapped containers if:

- Your organization requires containers to communicate only with approved internal services
- You need to meet compliance standards that mandate network isolation (such as SOC 2, ISO 27001, or PCI DSS)
- You want to prevent containers from leaking data or reaching unapproved external endpoints during builds or at runtime

How air-gapped containers work

`containersProxy` governs two distinct traffic paths:

- Image pulls (always enforced): Docker Desktop hardwires `http.docker.internal:3128` as the daemon's proxy in `daemon.json` at VM startup, so all `docker pull` and Compose pull operations always go through `containersProxy`, including any PAC file rules.
- Running container outbound traffic (opt-in): Docker Desktop intercepts container TCP connections and applies proxy rules only for ports listed in `transparentPorts`. Without it, running container traffic bypasses `containersProxy` entirely.

> [!IMPORTANT]
>
> If you configure a PAC file under `containersProxy`, the PAC file must return an appropriate proxy server to connect to the registries where your images are hosted.

Other considerations:

- If PAC file download fails, containers block requests to target URLs
- Hostname is available for ports 80 and 443, but only IP addresses for other ports

Prerequisites

Before configuring air-gapped containers, you must have:

- Enforce sign-in enabled to ensure users authenticate with your organization
- A Docker Business subscription
- Configured Settings Management with the `admin-settings.json` file to manage organization policies

Configure air-gapped containers

Add the container proxy to your `admin-settings.json` file. For example:

```
{
  "configurationFileVersion": 2,
  "containersProxy": {
    "locked": true,
    "mode": "manual",
    "http": "",
    "https": "",
    "exclude": [],
    "pac": "http://192.168.1.16:62039/proxy.pac",
    "transparentPorts": "*"
  }
}
```

Configuration parameters

The `containersProxy` setting controls network policies applied to `docker image pull` and, when `transparentPorts` is configured, running container outbound traffic:

Parameter	Description	Value
`locked`	Prevents developers from overriding settings	`true` (locked), `false` (default)
`mode`	Proxy configuration method	`system` (use system proxy), `manual` (custom)
`http`	HTTP proxy server	URL (e.g., `"http://proxy.company.com:8080"`)
`https`	HTTPS proxy server	URL (e.g., `"https://proxy.company.com:8080"`)
`exclude`	Bypass proxy for these addresses	Array of hostnames/IPs
`pac`	Proxy Auto-Configuration file URL	URL to PAC file
`transparentPorts`	Ports subject to proxy rules	Comma-separated ports or wildcard (`"*"`)

Configuration examples

Block all external access:

```
"containersProxy": {
  "locked": true,
  "mode": "manual",
  "http": "",
  "https": "",
  "exclude": [],
  "transparentPorts": "*"
}
```

Allow specific internal services:

```
"containersProxy": {
  "locked": true,
  "mode": "manual",
  "http": "",
  "https": "",
  "exclude": ["internal.company.com", "10.0.0.0/8"],
  "transparentPorts": "80,443"
}
```

Route through corporate proxy:

```
"containersProxy": {
  "locked": true,
  "mode": "manual",
  "http": "http://corporate-proxy.company.com:8080",
  "https": "http://corporate-proxy.company.com:8080",
  "exclude": ["localhost", "*.company.local"],
  "transparentPorts": "*"
}
```

Proxy Auto-Configuration (PAC) files

PAC files provide fine-grained control over container network access by defining rules for different destinations.

Basic PAC file structure

```
function FindProxyForURL(url, host) {
  if (localHostOrDomainIs(host, 'internal.corp')) {
    return "PROXY 10.0.0.1:3128";
  }
  if (isInNet(host, "192.168.0.0", "255.255.255.0")) {
      return "DIRECT";
  }
      return "PROXY reject.docker.internal:1234";
  }
```

General considerations

- `FindProxyForURL` function URL parameter format is `http://host_or_ip:port` or `https://host_or_ip:port`
- If you have an internal container trying to access `https://docs.docker.com//desktop/enterprise/hardened-desktop/air-gapped-containers` the Docker proxy service will submit docs.docker.com for the host value and https://docs.docker.com:443 for the URL value to `FindProxyForURL`, if you are using `shExpMatch` function in your PAC file as follows:

  ```
  if(shExpMatch(url, "https://docs.docker.com:443//desktop/enterprise/hardened-desktop/*")) return "DIRECT";
  ```

  `shExpMatch` function will fail, instead use:

  ```
  if (host == docs.docker.com && url.indexOf(":443") > 0) return "DIRECT";
  ```

PAC file return values

Return value	Action
`PROXY host:port`	Route through HTTP proxy at specified host and port
`SOCKS5 host:port`	Route through SOCKS5 proxy at specified host and port
`DIRECT`	Allow direct connection without proxy

Return value	Action
`PROXY reject.docker.internal:any_port`	Block the request completely

Advanced PAC file example

```
function FindProxyForURL(url, host) {
  // Allow access to Docker Hub for approved base images
  if (dnsDomainIs(host, ".docker.io") || host === "docker.io") {
    return "PROXY corporate-proxy.company.com:8080";
  }

  // Allow internal package repositories
  if (localHostOrDomainIs(host, 'nexus.company.com') ||
      localHostOrDomainIs(host, 'artifactory.company.com')) {
    return "DIRECT";
  }

  // Allow development tools on specific ports
  if (url.indexOf(":3000") > 0 || url.indexOf(":8080") > 0) {
    if (isInNet(host, "10.0.0.0", "255.0.0.0")) {
      return "DIRECT";
    }
  }

  // Block access to developer's localhost
  if (host === "host.docker.internal" || host === "localhost") {
    return "PROXY reject.docker.internal:1234";
  }

  // Block all other external access
  return "PROXY reject.docker.internal:1234";
}
```

Verify air-gapped container configuration

After applying the configuration, test that container network restrictions work:

Test blocked access:

```
$ docker run --rm alpine wget -O- https://www.google.com
# Should fail or timeout based on your proxy rules
```

Test allowed access:

```
$ docker run --rm alpine wget -O- https://internal.company.com
# Should succeed if internal.company.com is in your exclude list or PAC rules
```

Test proxy routing:

```
$ docker run --rm alpine wget -O- https://docker.io
# Should succeed if routed through approved proxy
```

Security considerations

- Network policy enforcement: Air-gapped containers work at the Docker Desktop level. Advanced users might bypass restrictions through various means, so consider additional network-level controls for high-security environments.
- Development workflow impact: Overly restrictive policies can break legitimate development workflows. Test thoroughly and provide clear exceptions for necessary services.
- PAC file management: Host PAC files on reliable internal infrastructure. Failed PAC downloads result in blocked container network access.
- Performance considerations: Complex PAC files with many rules may impact container network performance. Keep rules simple and efficient.

Next steps

- Explore Enhanced Container Isolation to further restrict what containers can do at runtime
- Understand how Docker Desktop handles host and container networking

---

□□□ #2
□ □□□ □□□ 2026-10-31 19:10:05 UTC
□ □□□ □□□ 2026-10-31 19:10:06 UTC