Distributive

DOCKER WORKER ↘

Updated: 6 Jan 2026

---

**Quickstart instructions for launching a DCP Docker Worker in the Global Network or in Private Compute Groups. Earned Compute Credits are deposited directly into your DCP bank account, which can be created in the DCP Portal at https://dcp.cloud**

---

## Docker Worker Overview

### Supported Platforms

Docker-compatible hosts, including Docker Desktop on macOS and Windows (64-bit; x86-64, arm64)

### Deployment Model

The Docker Worker is distributed as a pre-built container image (`distributivenetwork/dcp-worker`) and can be deployed on individual hosts using Docker or at scale using container orchestration platforms such as Kubernetes or OpenShift. For orchestrated deployments, Workers are typically launched via declarative configuration (e.g., YAML manifests or generated deployment templates) and managed using standard container lifecycle and scheduling mechanisms. A YAML manifest generator is available to simplify creating deployment templates for large-scale or private compute group deployments.

### Execution Model

The Docker Worker executes inside a Linux container managed by the host's container runtime. The container packages the same dcp-worker and sandboxed evaluator binaries used by Standalone Workers, providing a consistent and reproducible runtime environment across hosts.

Docker provides packaging, dependency isolation, and deployment portability. It does not alter the DCP Worker execution or sandbox security model; it adds an additional isolation boundary enforced by the container runtime.

Distributive

# Docker Worker Quickstart

Pull the latest DCP Docker Worker image from Docker Hub:

```bash
docker pull distributivenetwork/dcp-worker:latest
```

To view all supported command-line options and defaults, run the Worker with `--help`

```bash
docker run distributivenetwork/dcp-worker:latest --help
```

**Run in the Global Compute Group (Foreground, TUI)**
Run the Worker in the Global Compute Group and deposit earned Compute Credits into your DCP bank account (replace the account below with your own from dcp.cloud):

```bash
docker run -it distributivenetwork/dcp-worker:latest \
  --earnings-account=0x6aea918f84eaad8831599d3e3269d15ff81fb64c
```

Press `<Esc>` twice to exit the Worker

**Run in Private Compute Groups (Foreground, TUI)**
Run the Worker without joining the Global Compute Group and instead join one or more Private Compute Groups using a join key and secret:

```bash
docker run -it distributivenetwork/dcp-worker:latest \
  --earnings-account=0x6aea918f84eaad8831599d3e3269d15ff81fb64c \
  --no-global \
  --join demo,dcp
```

Replace `demo,dcp` with the appropriate group name and secret for your deployment.

### Run with a Persistent DCP Identity

To register the Worker in the DCP Portal and manage it under your account, provide a DCP identity keystore. Assuming `~/.dcp/id.keystore` exists on the host:

```bash
docker run -it \
  -v $HOME/.dcp/id.keystore:/id.keystore:ro \
  distributivenetwork/dcp-worker:latest \
  --dcp-identity=/id.keystore \
  --earnings-account=0x6aea918f84eaad8831599d3e3269d15ff81fb64c \
  --no-global \
  --join demo,dcp
```

The keystore is mounted read-only and used only for Worker identity and authentication.

There are many additional options, such as specifying allowed origins for routing data, functions, and results directly behind the firewall (e.g., for hospital genomics data processing), or setting target CPU and GPU loads to throttle device consumption, etc. Use `--help` to see the full list of options.

## Process and Isolation Model

All evaluator processes run inside the container under an unprivileged user and inherit no elevated privileges. Evaluators communicate exclusively with the Worker over container-local networking (loopback).

All sandbox security guarantees described in the Core DCP Worker Security Model apply unchanged. Docker adds an additional isolation boundary enforced by the container runtime (Linux namespaces and cgroups) but does not modify the Worker's execution or sandbox security model. Filesystem access is limited to explicitly mounted volumes, if any, and evaluator sandboxes operate entirely within the container boundary.

## Resource Scheduling

CPU, memory, and GPU resource allocation are governed by the container runtime and host operating system.

- CPU and memory limits may be enforced via container runtime configuration.

- GPU access is mediated through the host's GPU drivers and container runtime (e.g., NVIDIA Container Toolkit).
- The DCP Worker does not install kernel modules or modify host system configuration.

By default, the Worker will make all container-visible CPU cores and GPUs available for computation unless constrained by runtime configuration or Worker flags.

## Operational Characteristics

Docker Workers are typically used for:

- Cloud and data-center deployments
- Ephemeral or auto-scaled compute pools
- Environments favoring immutable infrastructure and declarative orchestration

They preserve the same core sandboxing and security properties as other DCP Worker variants, with containerization providing standardized packaging and deployment rather than altering the execution or security model.

*Happy computing*