Distributive

LINUX WORKER ↘

Updated: 6 Jan 2026

Quickstart instructions for launching a DCP Linux 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

---

Linux / Unix Worker Overview

Supported Platforms

Ubuntu Linux 20.04, 22.04, 24.04, 25.04, 25.10 (64-bit; x86-64, arm64)

Deployment Model

The Linux / Unix Worker is distributed as a `.deb` package or via Distributive's official APT repository, providing straightforward installation and update management. It can be deployed on individual machines or scaled across institutional fleets using standard automation and configuration management tools (shell scripts, Ansible, Puppet, etc.).

Execution Model

On Linux and Unix systems, the Standalone Worker runs continuously as a system-managed service under an unprivileged system user (typically `dcp`). The `dcp-worker` package installs a dedicated user and a `systemd` service (`dcp-worker.service`) responsible for lifecycle management. The Worker is never executed with root privileges during normal operation.

`systemd` handles:

- Starting the Worker at boot
- Restarting the Worker on failure
- Capturing stdout/stderr logs
- Enforcing basic process isolation and resource accounting

Distributive

Linux Worker Quickstart (APT)

Update the system:

```bash
sudo apt update && sudo apt upgrade -y
```

Install `dcp-worker`

```bash
wget -qO- https://apt.distributive.network/apt-setup.sh | bash
sudo apt-get install -y dcp-worker
```

After installation, the `dcp-worker` systemd service starts automatically and begins executing Jobs immediately.

To stop all DCP services:

```bash
sudo systemctl stop dcp-worker
```

Monitoring a Running Worker

Option A: Foreground Mode (Text UI)

Stop the background service if it is running:

```bash
sudo systemctl stop dcp-worker
```

Run the Worker manually as the unprivileged dcp user:

```bash
sudo --user=dcp /opt/dcp/bin/dcp-worker.sh
```

Press **Esc** twice to exit.

Example with custom options:

```bash
sudo --user=dcp /opt/dcp/bin/dcp-worker.sh \
  --dcp-identity=<id-private-key> \
  --earnings-account=<bank-account> \
  --no-global \
  --join <key>,<secret> \
  --utilization=1,1 \
  --cores=4,1
```

Option B: Background Mode (systemd)

Start the Worker:

```bash
sudo systemctl start dcp-worker
```

View logs from the last 5 minutes:

```bash
sudo journalctl -f --since="5 minutes ago" -u dcp-worker
```

Press **Ctrl+C** to exit log monitoring.

Configuring the Worker Service

Edit the systemd service file:

```bash
sudo vim /etc/systemd/system/dcp-worker.service
```

Modify the ExecStart line to include desired parameters:

```ini
ExecStart=/opt/dcp/bin/dcp-worker.sh --output=console --dcp-identity=<id-private-key>
--earnings-account=<bankAccount> --no-global --join <key>,<secret>
```

Apply changes (`Esc` then `:wq` then `Enter`) and restart the Worker:

```bash
sudo systemctl daemon-reload && sudo systemctl restart dcp-worker
```

Updating the Worker

Check for available updates:

```bash
apt list --upgradable | grep dcp-worker
```

Upgrade `dcp-worker`:

```bash
sudo apt update
sudo apt-get install --only-upgrade dcp-worker
```

Restart the Worker after updating:

```bash
sudo systemctl restart dcp-worker
```

Verify the installed version:

```bash
dcp-worker --version
```

There are several 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 Privilege Model

All evaluator processes run as child processes of the unprivileged `dcp-worker` service and inherit no elevated privileges or Linux capabilities. Evaluators communicate only with the Worker service over loopback TCP. No evaluator process accepts inbound network connections, accesses the host filesystem, or executes with administrative privileges. The runtime process hierarchy on Linux is:

```
systemd
 └─ dcp-worker (Node.js, unprivileged user)
 └─ dcp-evaluator (one per CPU core and GPU)
```

Resource Scheduling and Isolation

Process scheduling and resource allocation are entirely delegated to the host operating system. By default, the Worker will make all detected CPU cores and GPUs available for computation. Administrators may restrict resource usage via:

- Command-line flags (`-c cpuCount,gpuCount`)
- `systemd` service configuration (e.g., CPU quotas, cgroups)
- Standard OS-level scheduling and policy controls

Operational Characteristics

Linux / Unix Standalone Workers are typically used for:

- Institutional or enterprise deployments on desktops or servers
- Continuous background execution for CPU- and GPU-based distributed computation
- Scavenging idle compute cycles in multi-tenant environments

They preserve the same core sandboxing and security properties as other DCP Worker variants, while leveraging the host OS's service management and privilege model (`systemd`, unprivileged users) to manage lifecycle, isolation, and resource allocation.

Happy computing