Dotnet Optimized for HPC

- v1 - ami-0346c3e688659da71

v1 - ami-0346c3e688659da71

Propagate .NET HPC AMI

A performance-tuned Ubuntu 24.04 LTS image with **.NET 6, 8 and 10 SDKs** installed side by side, ready for compute-intensive .NET workloads. The system ships pre-tuned for high performance computing (Server GC, CPU governor, NUMA, huge pages, kernel and network tuning) and includes an optional containerized .NET app-hosting mode behind nginx-proxy with Let's Encrypt.

> Security is handled by **AWS Security Groups only**. There is no host firewall on this AMI — open just the ports you need in your Security Group.

Requirements

Resource	Minimum	Recommended
Instance type	t3.large (2 vCPU)	Compute-optimized c6i / c7i / hpc6a
RAM	4 GB	8 GB+ (more for large datasets)
Root volume	30 GB	30 GB+ (3 SDKs use ~3 GB)
Data volume (EBS)	optional	Attach for persistent app/job data

For real HPC throughput, pick a compute-optimized or HPC instance family and attach a second EBS volume for your data.

First boot

On first launch the instance auto-configures via `dotnet-hpc-firstboot.service`:

1. **No user-data** → applies the `throughput` HPC profile, mounts any attached data volume, sets .NET 8 as default, and does **not** start any container. SSH in and start building immediately.
2. **JSON user-data** → applies your settings and (optionally) deploys a containerized .NET app. Example user-data:

```
{
  "dotnet_version": "8",
  "hpc_profile": "throughput",
  "hugepages_mb": 0,
  "enable_app": true,
  "app_image": "your-registry/your-app:latest",
  "internal_port": 8080,
  "domain": "app.example.com",
  "enable_https": true,
  "letsencrypt_email": "admin@example.com",
  "admin_user": "admin",
  "admin_password": "ChangeMe123!"
}
```

To configure (or re-configure) interactively at any time:

```
sudo bash /opt/dotnet-hpc/configure-dotnet-hpc.sh
```

After configuring, open a new shell (or `source /etc/profile.d/dotnet-hpc.sh`) so the .NET environment variables load.

Using .NET

All three SDKs live under `/usr/share/dotnet` and `dotnet` is on the PATH.

```
dotnet --list-sdks         # show installed SDKs
dotnet --list-runtimes     # show installed runtimes
dotnet new console -o app  # uses the default SDK
```

Set the default SDK for new projects (writes a `global.json` template):

```
sudo dotnet-hpc default 10
```

> .NET 6 is provided for legacy compatibility and is past Microsoft's support window. Use .NET 8 or 10 for new work.

HPC tuning

The image applies Server GC, the `performance` CPU governor, NUMA settings, raised file/socket limits and network buffer tuning. Switch profiles anytime:

```
sudo dotnet-hpc tune throughput   # max compute throughput (default)
sudo dotnet-hpc tune latency      # low/steady latency for services
sudo dotnet-hpc tune balanced     # general purpose
```

Verify the environment and run a quick parallel benchmark:

```
dotnet-hpc status
dotnet-hpc bench 8        # Monte Carlo Pi across all vCPUs
```

Optional .NET large-page GC (reserve huge pages first, e.g. 2048 MiB):

```
# during configure, answer the "huge pages" prompt with a MiB value
```

Management commands

```
dotnet-hpc status            Show versions, tuning state and app stack
dotnet-hpc versions          List installed SDKs and runtimes
dotnet-hpc default <6|8|10>  Set default SDK
dotnet-hpc tune <profile>    Re-apply HPC profile
dotnet-hpc bench [6|8|10]    Run a compute benchmark
dotnet-hpc info              Show saved configuration
dotnet-hpc backup            Tar.gz the data volume to /root

# app stack (only if containerized hosting is enabled)
dotnet-hpc start | stop | restart | logs | update
```

Ports

Port	Purpose	When
22	SSH	Always (open to your IP only)

Port	Purpose	When
80	HTTP (nginx-proxy)	App hosting enabled
443	HTTPS (nginx-proxy)	App hosting enabled with Let's Encrypt
8080	App container (internal)	Never exposed; reached via the proxy

Open the relevant ports in your **Security Group**. Pure compute use needs only port 22.

HTTPS notes

- HTTPS uses nginx-proxy + acme-companion (Let's Encrypt).
- Let's Encrypt requires a **domain name**, not a bare IP. Point an A record at the instance's public IP before enabling HTTPS.
- The first certificate may take 1–2 minutes to issue after the first request.
- The app is protected by HTTP basic-auth using the admin credentials you set.
- Public IPs change on stop/start. Use an Elastic IP or a domain for stable access.

Backup

```
sudo dotnet-hpc backup
```

Writes `/root/dotnet-hpc-backup-<timestamp>.tar.gz` of the data volume. For durable backups, copy that archive to Amazon S3, or take an EBS snapshot of the attached data volume from the AWS console / CLI.

Support

Documentation: https://docs.propagate.solutions 30-day money-back guarantee.