HyperSDK

Training & Team Onboarding

Structured learning paths, hands-on guides, and reference materials to get your team productive with HyperSDK in hours, not weeks.

30-Minute Quick Start -- CLI Reference Card -- Self-Paced Learning

Learning Path

Structured progression from beginner to advanced, with estimated hours for each level.

Beginner (2-4 hours)

- Install HyperSDK and hyperctl
- Configure first provider
- Run your first VM export
- Explore the web dashboard
- Understand job lifecycle
- Read output and logs

Intermediate (4-8 hours)

- Multi-provider configuration
- Backup scheduling with cron
- Webhook notifications setup
- Format conversion workflows
- API usage and authentication
- Performance tuning basics

Advanced (8-16 hours)

- Hexagonal architecture internals
- Custom provider development
- Kubernetes operator deployment
- Grafana dashboard customization
- DR planning and testing
- Contributing to HyperSDK

30
Minutes to First Export

4
Hours to Productive Use

16
Hours to Full Mastery

10+
Hands-On Examples

Teams consistently report going from zero to running production exports within their first day. The CLI is intuitive and the documentation covers every use case.

Getting Started

Install, configure, and run your first export in 30 minutes.

1. **Install HyperSDK (5 minutes)**

   Clone the repository and build from source. Requires Go 1.24+. Run `make build` to compile the daemon and `hyperctl` CLI.

2. **Configure a Provider (5 minutes)**

   Create `hypersdk.yaml` with your provider credentials. Start with one provider (e.g., vSphere or AWS). Run `hyperctl provider test` to verify connectivity.

3. **Start the Daemon (2 minutes)**

   Run `make run` for development or `systemctl start hypervisord` for production. The daemon starts on port 8080 by default.

4. **Discover VMs (3 minutes)**

   Use `hyperctl list` to discover available VMs from your configured provider. Browse by name, ID, or tag to find your target VM.

5. **Run Your First Export (10 minutes)**

   Submit an export job with `hyperctl submit --provider vsphere --vm my-vm --format ova --output ./exports/`. Monitor progress in real-time.

6. **Explore the Dashboard (5 minutes)**

   Open `http://localhost:8080` in your browser to access the React dashboard. View jobs, providers, and system metrics in a visual interface.

```
# Complete quick start in 6 commands:
git clone https://github.com/hypersdk/hypersdk && cd hypersdk
make build
cp examples/hypersdk.yaml.example hypersdk.yaml   # Edit with your credentials
make run &
hyperctl provider test vsphere
hyperctl submit --provider vsphere --vm my-vm --format ova --output ./exports/
```

Key Concepts

Core abstractions: providers, jobs, schedules, webhooks, and the API architecture.

Providers

Providers are adapters to cloud platforms (vSphere, AWS, Azure, GCP, Proxmox, OCI, etc.). Each implements the `ComputeProvider` and `ExportProvider` port interfaces for a consistent API across all clouds.

Jobs

Jobs represent units of work: export, convert, migrate, or backup. Each job has a lifecycle (pending, running, completed, failed) managed by the `JobManager` port. Jobs are persistent and resumable.

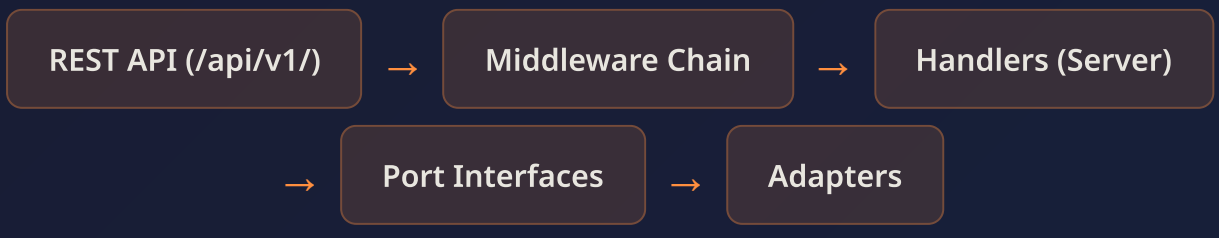
Schedules

Schedules define recurring job execution via cron expressions. Managed by the `ScheduleManager` port. Schedules support timezone awareness, time windows, and priority levels.

Webhooks

Webhooks deliver event notifications to external systems. Configure endpoints for job lifecycle events (started, completed, failed). Supports Slack, Discord, and generic HTTP endpoints.

API Architecture



The API follows hexagonal architecture. Routes are registered via `registerVersionedRoute()` at `/api/v1/` with legacy path support. The middleware chain processes requests through: logging, request ID, CORS, security headers, authentication, rate limiting, and request size limiting.

Layer	Location	Purpose
Domain Types	`internal/domain/`	70+ canonical types (Job, Schedule, Provider, etc.)
Port Interfaces	`internal/ports/`	13+ interfaces (JobManager, ScheduleManager, etc.)
Adapters	`daemon/`, `providers/`	Concrete implementations for each provider
API Server	`daemon/api/server.go`	HTTP handlers, routes, middleware

CLI Mastery

hyperctl command reference card -- the essential commands for daily operations.

Job Management

Command	Description	Example
`hyperctl submit`	Submit a new export/migration job	`hyperctl submit --provider aws --vm i-abc --format qcow2`
`hyperctl query`	Get detailed job status and progress	`hyperctl query --id j-abc123`
`hyperctl list`	List jobs with optional filters	`hyperctl list --status=running --provider=vsphere`
`hyperctl cancel`	Cancel a running or pending job	`hyperctl cancel --id j-abc123`

Provider & Discovery

Command	Description	Example
`hyperctl provider list`	List configured providers	`hyperctl provider list`
`hyperctl provider test`	Test provider connectivity	`hyperctl provider test vsphere`
`hyperctl grep`	Search VMs by name pattern	`hyperctl grep "prod-web-*"`

Advanced Operations

Command	Description	Example
`hyperctl hyper2kvm`	Convert Hyper-V VMs to KVM format	`hyperctl hyper2kvm --input vm.vhdx --output vm.qcow2`
`hyperctl schedule`	Manage recurring job schedules	`hyperctl schedule create --cron "0 2 * * *"`
`hyperctl health`	Check daemon health status	`hyperctl health`
`hyperctl rg`	Search across VM metadata	`hyperctl rg "ubuntu" --provider aws`

Resources

Documentation, examples, community channels, and support resources for your team.

📖

Documentation

- **Architecture Guide:** `docs/architecture/ARCHITECTURE.md`
- **Cloud Providers:** `docs/cloud-providers/`
- **Testing Guide:** `docs/testing/testing-guide.md`
- **API Reference:** `/api/v1/` endpoints
- **Presentations:** `docs/client-presentations/`

💻

Examples

- **Examples Index:** `examples/EXAMPLES_INDEX.md`
- **Quick Start:** `examples/README.md`
- **AWS Migration:** `docs/cloud-providers/aws-migration-guide.md`
- **OCI Integration:** `docs/cloud-providers/oci-integration.md`
- **Proxmox Setup:** `docs/cloud-providers/proxmox-integration.md`

🤝

Community

- **GitHub:** Issues, discussions, and pull requests
- **Changelog:** `CHANGELOG.md` for release notes
- **Contributing:** Fork, branch, test, PR workflow

🏳️

Support Channels

- **GitHub Issues:** Bug reports and feature requests
- **Discussions:** Questions and best practices

- **Code of Conduct:** Welcoming and inclusive community
- **License:** Apache 2.0 open source

- **Documentation:** Self-service reference
- **Test Results:** `docs/test-results.md`
- **Presentations:** Shareable slide decks for stakeholders

> *HyperSDK is built with Go 1.24 using hexagonal architecture. The codebase is clean, well-tested, and documented. New contributors can understand the structure and start contributing within a day.*

Start Your Team's Journey Today

From installation to production exports in 30 minutes. Comprehensive documentation, hands-on examples, and a welcoming community make HyperSDK the easiest way to manage multi-cloud VM operations.