Datomic Cloud Documentation

Search

- Home ›
- ›
- Local Dev and CI with dev-local
- Support
- Forum

What is Datomic?

- Data Model
- Architecture
- Supported Operations
- Programming with Data and EDN

Local Dev and CI

Cloud Setup

- Start a System
- Configure Access
- Get Connected

Tutorial

- Client API
- Assertion
- Read
- Accumulate
- Read Revisited
- Retract
- History

Client API

Videos

- AWS Setup
- Edn
- Datalog
- Datoms
- HTTP Direct
- CLI tools

Schema

- Defining Attributes
- Schema Reference
- Changing Schema

- Data Modeling
- Schema Limits

Transactions

- Processing Transactions
- Transaction Data Reference
- Transaction Functions
- ACID
- Client Synchronization

Query and Pull

- Executing Queries
- Query Data Reference
- Pull
- Index-pull
- Raw Index Access

Time in Datomic

- Log API
- Time Filters

Ions

- Ions Tutorial
- Ions Reference
- Monitoring Ions

Analytics Support

- Configuration
- Connecting
- Metaschema
- SQL CLI
- Troubleshooting

Analytics Tools

- Metabase
- R
- Python
- Jupyter
- Superset
- JDBC
- Other Tools

Operation

- Planning Your System
- Start a System
- AWS Account Setup
- Access Control
- CLI Tools
- Client Applications
- High Availability (HA)

Hide All Examples

Local Dev and CI with dev-local

Concept

With Datomic dev-local you can develop and test applications with minimal connectivity and setup. Get the dev-local library, add it to your classpath, and you have full access to the Client API (both synchronous and asynchronous flavors). This allows you to:

- Develop and test Datomic Cloud applications without connecting to a server and without changing your application code.
- Create small, single-process Datomic applications and libraries.

Dev-local is available at no cost. Note that dev-local is not redistributable. If you make Datomic apps/libs for others to use they must get a copy of dev-local themselves.

Getting Setup

Get the latest version (0.9.184) of dev-local and unzip it. From the unzip directory, install dev-local in your local repository with:

```
mvn org.apache.maven.plugins:maven-install-plugin:3.0.0-M1:install-file -Dfile=dev-local-0.9.184.jar
```

By default, dev-local stores all databases under a common storage directory. To specify this directory, create a `.datomic/dev-local.edn` file in your home directory, containing a map with `:storage-dir` and an absolute path:

```
{:storage-dir "/full/path/to/where/you/want/data"}
```

Using dev-local

There are two steps to use dev-local: add the dev-local library to your classpath and create a local client.

To add dev-local to your classpath, add a `com.datomic/dev-local` entry to your deps.edn file.

```
{com.datomic/dev-local
 {:mvn/version "<dev-local-version>"}}
```

The dev-local dependency is all you need for using dev-local.

If you intend to use dev-local in a system that also uses Datomic Cloud (i.e. using `import-cloud` or `divert-system`), you must also include the Datomic `client-cloud` dependency in your project.

There are two ways to get a local client:

- you can explicitly request a local client
- you can *divert* requests for Datomic Cloud clients to use local storage for development and testing

To explicitly request a local client, pass a map to `d/client` with

- `:server-type :dev-local`
- a `:system` name,

```
(require '[datomic.client.api :as d])
(def client (d/client {:server-type :dev-local
                       :system "dev"}))
```

⇧

To divert an existing Datomic Cloud system to dev-local, call divert-system:

```
(require '[datomic.dev-local :as dl])
(dl/divert-system {:system "production"})

;; existing requests for Cloud system will be served locally!
(def client (d/client {:system "production"
                       :server-type :ion
                       :region "us-east-1"
                       :endpoint "http://entry.production.us-east-1.datomic.net:8182/"}))
```

⇧

You can also use import-cloud to import data from Datomic Cloud to local storage.

If you are new to Datomic, you can now work through the tutorial.

Durability

dev-local stores data to your local filesystem, in directories under the `:system` you specify when creating a dev-local client.

Each database will store transactions in a directory named `<storage-dir>/<system-name>/<database-name>`. You can "backup" or "restore" a dev-local database simply by copying the database directory.

API

Most usage of dev-local should be via portable Client API calls. Capabilities that are specific to dev-local are available through the `datomic.dev-local` namespace:

- divert-system to develop and test Datomic Cloud applications against dev-local
- release-db to release memory used by a database
- import-cloud to import a Cloud database for dev-local use

All dev-local API calls take a single arg-map argument.

```
(import-cloud arg-map)
(divert-system arg-map)
(release-db arg-map)
```

⇧

divert-system

Diverts subsequent d/client calls for system to local storage. arg-map has the following keys:

Key	Value	Required
:system	the system to divert	yes
:storage-dir	overrides :storage-dir in ~/.datomic/dev-local.edn	

release-db

Closes the connection to a database and releases the memory used by it. Values of that database can no longer be used. arg-map has the following keys:

Key	Value	Required
:system	system name	yes
:db-name	database name	yes

import-cloud

Import a Datomic Cloud database into dev-local. arg-map has the following keys:

Key	Value	Required
:source	arg map for (Cloud) d/client merged with arg map for d/connect	yes
:dest	arg map for (Cloud) d/client merged with arg map for d/connect	yes
:filter	filter spec limits datoms to import	

A filter spec has the following keys, all optional:

Key	Value
:before	t - include only txes before t, exclusive
:since	t - include only txes after t, inclusive

Key	Value
:include-attrs	map of attr -> filter
:exclude-attrs	vector of attrs to be excluded

`:include-attrs` keys are either fully-qualified attribute names or a namespaced keyword with * as a name (includes all attrs in namespace, e.g. `:order/*`).

`:include-attrs` filters are maps with optional `:before` and `:since` keys limiting the time range for attributes as per before/since above.

Schema attributes are always included. When a filter spec is present:

- All other attributes must be included explicitly.
- Excludes take precedence over includes.

You can re-import to get more recent data if you have not transacted locally. You cannot import to a database that currently has an open connection.

The following example imports a customer database, including

- all schema
- all history of customers (except their secrets!)
- orders since May 2020

```
(dl/import-cloud
 {:source {:system "customers"
           :db-name "customers"
           :server-type :ion
           :region "us-east-1"
           :endpoint "http://entry.inventory.us-east-1.datomic.net:8182/"}
  :dest {:system "production-imports"
         :server-type :dev-local
         :db-name "customers-and-recent-orders"}
  :filter {:include-attrs
           {:customer/* {}
            :order/* {:since #inst \"2020-05\"}}
           :exclude-attrs [:customer/secret]}})
```

⇧

Performance Limits

- dev-local is in-process with your application code, and has all the tradeoffs (vs. a server or cluster) that this implies.
- dev-local requires 32 bytes of JVM heap per datom. You should plan your application with this in mind, while also leaving memory for your application's use.
- dev-local relies on OS page caching for performance, so leave some RAM available (i.e. not allocated to JVM heap) for this.

dev-local is best suited for small, single process applications. For larger projects, create a Datomic Cloud system.

Change Log

2020/07/24 - 0.9.184

- Improvement: better error message when calling a function that is not in the :allow list of datomic/ion-config.edn

2020/07/21 - 0.9.183

- Update: use latest Client API

2020/07/17 - 0.9.180

- Enhancement: Reread the deps-local.edn config file when creating a client.
- Enhancement: Provide better feedback while loading and importing databases.
- Enhancement: Update client documentation about connecting to dev-local systems

2020/07/10 - 0.9.172

Initial Release