Datomic Cloud Documentation

Search

Hide All Examples

How To

Table of Contents

Install the AWS CLI

Install the AWS Command Line Interface.. Make sure you have version 1.11.170 or later. If you are unfamiliar with managing Python environments, first try using the bundled AWS CLI installer. You can check your version using:

```
aws --version
```

NOTE: Running the `datomic-socks-proxy` script with an earlier version of the AWS CLI will result in a generic error message and failure of the script.

If you require multiple versions of the AWS CLI to be installed, then please investigate using Python's virtual environments, or the third-party virtualenv solution.

Manage AWS Access Keys for Datomic

The Datomic CLI tools and ion deployment tools require that you have a set of AWS access key with permissions to access Datomic. To create these keys, you must have an authorized IAM user. With this user you can then:

1. Create access keys
2. Add access keys to your environment

Datomic supports the use of named profiles as a credentials source. Additional information about IAM Users and access keys can be found in the AWS Security Credentials documentation .

Install Clojure CLI

Datomic's tools use the Clojure CLI, version 1.10.1.478 or later. You can use `clojure -Sdescribe` to check your version of the Clojure CLI.

The Clojure Getting Started page has instructions for installing and upgrading Clojure.

Install ion-dev Tools

The Datomic ion-dev tools should be installed via an alias in your user deps.edn file, which is normally located at `$HOME/.clojure/deps.edn`.

To install the tools, add the `datomic-cloud` maven repo under your `:mvn/repos` key:

```
"datomic-cloud" {:url "s3://datomic-releases-1fc2183a/maven/releases"}
```

Then add an `:ion-dev` entry under your `:aliases` key with the latest version of ion-dev:

```
:ion-dev
{:deps {com.datomic/ion-dev {:mvn/version "0.9.247"}}
 :main-opts ["-m" "datomic.ion.dev"]}
```

⇧

The ions tutorial includes a complete .clojure/deps.edn example.

Note that `ion-dev` configures logging to stderr via slf4j-simple. This is probably adequate for most scenarios, but you of course have the full power of SLF4J at your disposal.

Install ion Library

The Datomic ion library should be installed in the deps.edn for each ion project:

Add the `datomic-cloud` maven repo under your `:mvn/repos` key:

```
"datomic-cloud" {:url "s3://datomic-releases-1fc2183a/maven/releases"}
```

Then add the latest version of ion-dev to your `:deps`:

```
com.datomic/ion {:mvn/version "0.9.35"}
```

Delete Ion lambdas

You do **not** need to delete the AWS Lambdas managed by Datomic Ions in order to control cost. AWS Lambda pricing is entirely usage-based, so if you do not invoke Lambdas, they cost nothing. If you want to delete these Lambdas anyway, you can push and deploy an application with an empty `lambdas` map in ion-config.edn.

Control Shell Scripts

Some Datomic CLI tools (e.g. the client access script) continue to run in the foreground once you launch them. For interactive use, the easiest way to manage such tools is simply to kill them with Ctrl-C when they are no longer needed.

If you are building automation around scripts, you can of course use all the ordinary Unix facilities, e.g.

- You can use `pkill [script-name]` to kill a script from another terminal.
- You can use e.g. `nohup [script-command] [script-args] \~` to launch a script in the background, directing its output to `nohup.out`.

Find Datomic System Name

Every Datomic system resides in a single AWS region, and has a unique system name within that region. The system name is the Stack Name used to launch the Datomic storage stack.

If you did not start Datomic yourself, you can find the available systems in a region two ways:

- Utilize the `datomic cloud list-systems` command.

or

- Browse to CloudFormation console.

Find Datomic Nodes

You can list the names of all running Datomic nodes using:

- Utilize the `datomic system list-instances <system>` command, replacing `<system>` with your system name.

or

- the following AWS CLI command, replacing `[region]` with the region you want to query:

```
aws ec2 describe-instances --region [region] --filters "Name=tag-key,Values=datomic:tx-group" "Nam
```

Find Compute Group Name

- If you start a Datomic system with split stacks, then it has one or more compute groups. A compute group's name is the name of the CloudFormation stack you used to create the group.
- If you start a system using the AWS Marketplace template, Datomic generates a unique name for your compute stack based on your system name, of the form `$(SystemName)-Compute-$(GeneratedId)`.

You can:

- Utilize the `datomic system list-instances <system>` command, replacing `<system>` with your system name.

or

- browse all your CloudFormation stacks in the AWS CloudFormation console.

Check System Topology

All Datomic systems include one or more CloudFormation stacks. To check whether a system uses the Solo or Production topology either:

- Utilize the `datomic cloud list-systems` command.

or

- Find your system's primary compute group in the CloudFormation console.
- If the CloudFormation template has a `DatomicProductionCompute` key in its Outputs, the system is running the Production topology. If the key is absent, the system is running Solo.

Find Datomic Application Name

Every Datomic compute group has an associated application for ion deployments. You specify the application name when you create a compute group.

If you did not start the compute group, you can find the application name by browsing to your compute group in the AWS CloudFormation console and looking for the `ApplicationName` value in the Parameters tab.

Find System S3 Bucket

Every Datomic system has its own S3 bucket for configuration and data storage.

You can find this S3 bucket by browsing to the Outputs tab of your storage stack in the AWS Console and looking for the the `S3DatomicArn` output.

Find Ion Code Bucket

All Datomic systems in a region share a common S3 bucket for ion deployment. This bucket has a `datomic:code` AWS tag.

You can find this bucket by creating a resource group in the AWS Console:

- make sure you are working in the region you want to query
- create a tag based resource group
- select a resource type of `AWS::S3::Bucket`
- search for a tag named `datomic:code`
- give you group a name, e.g. `DatomicCodeBucket`
- create your group

Upgrade Base Schema

Every Datomic database starts with a base schema, whose identifiers are prefixed with `:db`. When the Datomic team enhances the base schema, all databases created with new versions of Datomic get the enhancements automatically.

Existing databases do **not** automatically incorporate enhancements to the base schema. In order to upgrade existing databases to use new base schema, you must first upgrade **all** compute groups (primary and query) to the minimum version of Datomic required for the base schema features you want:

Feature	Minimum Datomic Version	Released
tuples	480-8770	June 27, 2019
attribute predicates	480-8770	June 27, 2019
entity specs	480-8770	June 27, 2019
db.type/symbol	480-8770	June 27, 2019

One you have upgraded all compute groups, you can upgrade an existing database to use the latest base schema by passing the `:upgrade-schema` action to `administer-system`. For example:

```
(d/administer-system client {:db-name "movies"
                             :action :upgrade-schema})
```

Upgrading a schema is an idempotent operation, so it will not harm your data to call it more than once. That said, you should treat `administer-system` as a potentially expensive operation. Call it only when you actually need it, not before every call to `connect`.

> Once you have used a base schema feature in a particular database, do **not** run any compute group for that system on versions of Datomic older than the features used. (See table above.)

Update a CloudFormation Parameter

Datomic's CloudFormation parameter settings are carefully planned to adhere to AWS best practices and provide a robust and performant system. You should change CloudFormation parameters only:

- when the Datomic documentation specifically instructs you to do so
- when the Datomic support team specifically instructs you to do so

To update a CloudFormation parameter:

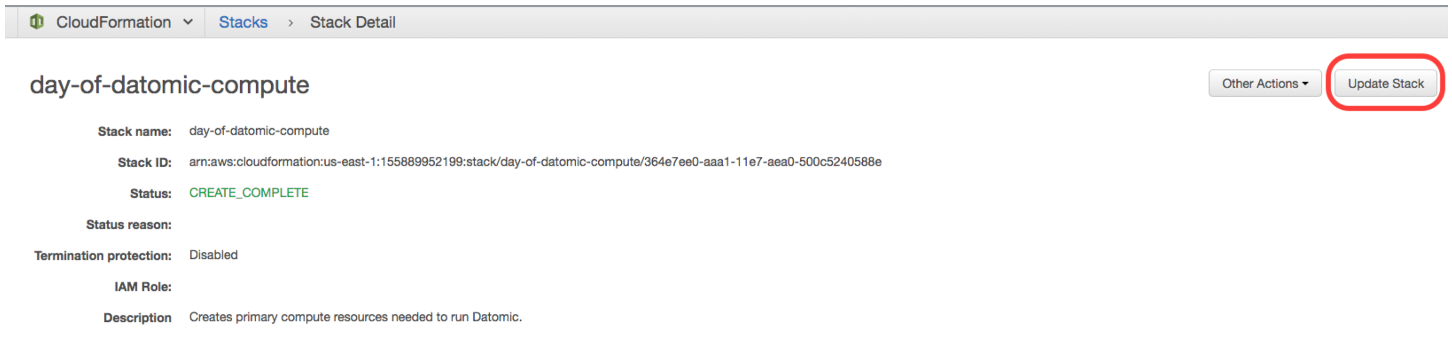
- Select your stack via the checkbox or radio button.
- Click the "Actions" button and select "Update Stack"
- On the "Select Template" screen, choose "Use current template"
- On the "Specify Details" screen, change your parameter(s).
- On the "Options" screen, leave all options unchanged.
- On the "Review" screen, click the checkbox stating "I acknowledge that AWS CloudFormation might create IAM resources with custom names."

Convert from Solo to Production

To update your system from Solo to Production topology:

- make sure you have raised your i3.large instance limit
- click the name of your solo stack in

CloudFormation console. On the stack page, Click "Update Stack":



day-of-datomic-compute

Stack name:	day-of-datomic-compute
Stack ID:	arn:aws:cloudformation:us-east-1:155889952199:stack/day-of-datomic-compute/364e7ee0-aaa1-11e7-aea0-500c5240588e
Status:	CREATE_COMPLETE
Status reason:	
Termination protection:	Disabled
IAM Role:	
Description	Creates primary compute resources needed to run Datomic.

On the following page, select "Specify an Amazon S3 template URL:" and enter the CloudFormation template URL for the latest production template you wish to upgrade to (see Release page for the latest production template) and click "Next".

- On the "Specify Details" screen, leave all options unchanged.
- On the "Options" screen, leave all options unchanged.
- On the "Review" screen, click the checkbox stating "I acknowledge that AWS CloudFormation might create IAM resources with custom names."

Wait for the template to report UPDATE_COMPLETE. This can take five or more minutes. You can refresh the CloudFormation dashboard to see progress. In the EC2 Dashboard you'll note that two i3.large instances with your stack name are running and a t2.small instance has been terminated. Your production stack is now ready for use.