Deployment

> **ALPHA STATUS**
>
> Chroma Server is currently in Alpha. We are working hard to move Chroma from an in-memory single-process oriented library to a distributed production-grade DB!
>
> - [x] Alpha ← Currently
> - [ ] Technical Preview - ~1 month away, powered by a completely new backend
> - [ ] Full production
> - [ ] GA - General Availability

Deployment

You can also deploy Chroma on a long-running server, and connect to it remotely.

There are many possible configurations, but for convenience we have provided a very simple AWS CloudFormation template to experiment with deploying Chroma to EC2 on AWS.

Hosted Chroma

We want to offer hosted Chroma, and we need your help.

Fill out the survey to jump the wait-list. Coming Q3 2023.

30 second survey

Simple AWS Deployment

⚠ Chroma and its underlying database need at least 2gb of RAM, which means it won't fit on the 1gb instances provided as part of the AWS Free Tier. This template uses a `t3.small` EC2 instance, which costs about two cents an hour, or $15 for a full month. If you follow these instructions, AWS will bill you accordingly.

⚠ This basic stack doesn't support any kind of authentication; anyone who knows your server IP will be able to add and query for embeddings. To secure this endpoint, you'll need to put it behind **AWS API Gateway** or add your own authenticating proxy.

⚠ By default, this template saves all data on a single volume. When you delete or replace it, the data will disappear. For serious production use (with high availability, backups, etc.) please read and understand the CloudFormation template and use it as a basis for what you need, or reach out to the Chroma team for assistance.

Step 1: Get an AWS Account

You will need an AWS Account. You can use one you already have, or **create a new one**.

Step 2: Get credentials

For this example, we will be using the AWS command line interface. There are **several ways** to configure the AWS CLI, but for the purposes of these examples we will presume that you have **obtained an AWS access key** and will be using environment variables to configure AWS.

Export the `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` environment variables in your shell:

```
export AWS_ACCESS_KEY_ID=****************
export AWS_SECRET_ACCESS_KEY=**********************
```

You can also configure AWS to use a region of your choice using the `AWS_REGION` environment variable:

```
export AWS_REGION=us-east-1
```

Step 3: Run CloudFormation

Chroma publishes Cloudformation templates to S3 for each release.

To launch the template using AWS CloudFormation, run the following command line invocation:

```
aws cloudformation create-stack --stack-name my-chroma-stack --template-url
```

```
https://s3.amazonaws.com/public.trychroma.com/cloudformation/latest/chroma.cf.json
```

Replace `--stack-name my-chroma-stack` with a different stack name, if you wish.

Wait a few minutes for the server to boot up, and Chroma will be available! You can get the public IP address of your new Chroma server using the AWS console, or using the following command:

```
aws cloudformation describe-stacks --stack-name my-chroma-stack --query
'Stacks[0].Outputs'
```

Step 4: Customize the Stack (optional)

The CloudFormation template allows you to pass particular key/value pairs to override aspects of the stack. Available keys are:

- `InstanceType` - the AWS instance type to run (default: `t3.small`)
- `KeyName` - the AWS EC2 KeyPair to use, allowing to access the instance via SSH (default: none)

To set a CloudFormation stack's parameters using the AWS CLI, use the `--parameters` command line option. Parameters must be specified using the format `ParameterName={parameter},ParameterValue={value}` .

For example, the following command launches a new stack similar to the above, but on a `m5.4xlarge` EC2 instance, and adding a KeyPair named `mykey` so anyone with the associated private key can SSH into the machine:

```
aws cloudformation create-stack --stack-name my-chroma-stack --template-url
https://s3.amazonaws.com/public.trychroma.com/cloudformation/latest/chroma.cf.json
\
    --parameters ParameterKey=KeyName,ParameterValue=mykey \
                 ParameterKey=InstanceType,ParameterValue=m5.4xlarge
```

Step 5: Configure the Chroma Library

When you launch the Chroma client library to actually use Chroma, all you need to do is configure it to use the server's IP address and port `8000` . You can do this in two ways:

Using Environment Variables

```
export CHROMA_API_IMPL=rest
export CHROMA_SERVER_HOST=<server IP address>
export CHROMA_SERVER_HTTP_PORT=8000
```

In Code

```
import chromadb
from chromadb.config import Settings
chroma = chromadb.HttpClient(host=<server IP address>, port=8000)
```

Step 6: Clean Up (optional).

To destroy the stack and remove all AWS resources, use the AWS CLI `delete-stack` command.

```
aws cloudformation delete-stack --stack-name my-chroma-stack
```

⚠ This will destroy all the data in your Chroma database, unless you've taken a snapshot or otherwise backed it up.

Troubleshooting

If you get an error saying `No default VPC for this user` when creating `ChromaInstanceSecurityGroup`, head to **AWS VPC section** and create a default VPC for your user.

Edit this page