# **Logjam.Next** *Installation Guide*

This guide is intended for Users & Developers of Logjam who need to install the required software to get Logjam running in a production-like environment. It describes the installation process, how to start necessary background services, and troubleshooting tips for getting started.

If you are a Logjam User who needs to make log queries, please see the ***User Guide*** for a tutorial on the Webpage and Command Line Interface (CLI). If you are a Logjam Developer who needs to modify the system, please see the ***Developer Guide*** for additional installation steps and information on the codebase.

## **Supported Platform**

Our only target platform is Ubuntu 18.04. Other UNIX-based operating systems are likely to work but have not been tested.

## **Hardware Considerations**

While there are no firm minimum specs, the following are considerations that should be made when provisioning hardware to run Logjam.

|  |  |
| --- | --- |
| CPU | The Logjam ingest script is designed to parallelize based on available CPU cores. More cores will generally improve performance significantly up to a certain point. |
| Storage | Based on a rough estimate of the input directory size and the approximate portion of relevant StorageGRID data, the storage requirement of the Elasticsearch index may eventually be as high as 50 Tb. See below for some guidelines on scaling Elasticsearch to potentially spread out this burden. In any case a very large volume will be needed to hold the Elasticsearch index. |
| RAM | Elasticsearch makes heavy use of memory to store a large portion of the index that can be accessed quickly. By design, Elasticsearch will consume more RAM than there is physically available and rely on the system to swap to some degree. More RAM available should mitigate this some and improve search times. |

In addition to the above, we recommend making a ramdisk (tmpfs) available to use as scratch space for the ingest script. This space should be large enough to accommodate multiple workers unzipping at once but the exact size needed cannot be predicted reliably. On some Linux systems, /tmp is a ramdisk by default but if not, one can be created by adding a line to /etc/fstab:

|  |
| --- |
| tmpfs /mnt/ramdisk tmpfs rw,size=2G 0 0 |

## **Installing Dependencies**

All commands are from the root of the git repository.

Install docker, docker-compose, & python3-venv:

|  |
| --- |
| sudo apt-get install docker.io docker-compose python3-venv p7zip-full |

\*Note that the package for Docker is not called ‘docker’ in the Ubuntu repositories.

Add your user to the docker group (or else run all docker commands as root which is not recommended):

|  |
| --- |
| sudo usermod -aG docker $(whoami) |

You will need to re-log for this to take effect.

## **Start Elasticsearch and Logjam UI**

Build Logjam-ui container:

|  |
| --- |
| docker-compose build |

This will download external JavaScript files which become part of the built image.

Create a data folder for Elasticsearch and Logjam. This folder will require a lot of available storage as the index grows.

|  |
| --- |
| mkdir ./data  mkdir ./data/elasticsearch/ && sudo chown 1000:1000 ./data/elasticsearch |

Start services (run from project root folder):

|  |
| --- |
| docker-compose up -d |

This will create a network with two containers: one for Elasticsearch, and one for our frontend: Logjam-ui. The containers will run in the background and can be stopped at any time by running docker-compose down from the same directory. Stopping them this way will also cleanup docker resources (such as network interfaces) which are no longer needed. Alternatively, plain docker commands (ps, run, stop) can be used to control the containers if desired.

Depending on your system, it may be necessary to increase the max\_map\_count option. This has to do with how much memory Elasticsearch is able to allocate.

|  |
| --- |
| sysctl -w vm.max\_map\_count=262144 |

For other issues related to running Elasticsearch in Docker, see the official Elasticsearch install guide [here](https://www.elastic.co/guide/en/elasticsearch/reference/current/docker.html). Additionally, Elasticsearch has the following recommendations for security and reliability in a production environment: [here](https://www.elastic.co/guide/en/elasticsearch/reference/current/docker.html#docker-prod-prerequisites).

## **Scaling Elasticsearch**

Elasticsearch provides built in options for scaling horizontally to multiple nodes. This is generally recommended when the index size is large since Elasticsearch can keep different “shards” of the index on different hardware nodes. We did a basic test scaling to three nodes but did not find a performance improvement in our case, probably due to increased i/o demands. However, Elasticsearch scaling is agreed among the community to be a big “your mileage may vary” situation so we have provided a framework of the configuration to achieve scaling.

To setup a multi-node cluster, you will need to clone the repository on each node. Next edit the file at src/elasticsearch/elasticsearch.yml (note the highlighted lines should be uncommented and updated to reflect the cluster deployment):

|  |
| --- |
| cluster.name: logjam  # Must bind to all local interfaces to work inside docker  network.bind\_host: 0.0.0.0  # Comment this line for multi-node deployments  discovery.type: single-node  # Set the IP of one or more nodes that \*could\* be the master.  # If more than one, they will sort it out among themselves.  cluster.initial\_master\_nodes:  - 152.14.85.174  # Set to the external IP of \*this\* host (one that other nodes can reach it by)  network.publish\_host: 152.14.85.174  # List ALL nodes in the cluster by their externally accessible IP  discovery.seed\_hosts:  - 152.14.85.174  - 152.14.85.175 |

*On each node:*

* Set the publish host to its external ip address or hostname
* Set the seed hosts to include every node in the cluster (this is the same on all)
* Set the master nodes to one or more nodes you want to act as master. This list **must** match on every node.

Then startup elasticsearch on each node like so:

|  |
| --- |
| docker-compose up -d elasticsearch |