Module 3

Purwadhika
Digital Technology School

Full Stack Web Development

Advanced Topic

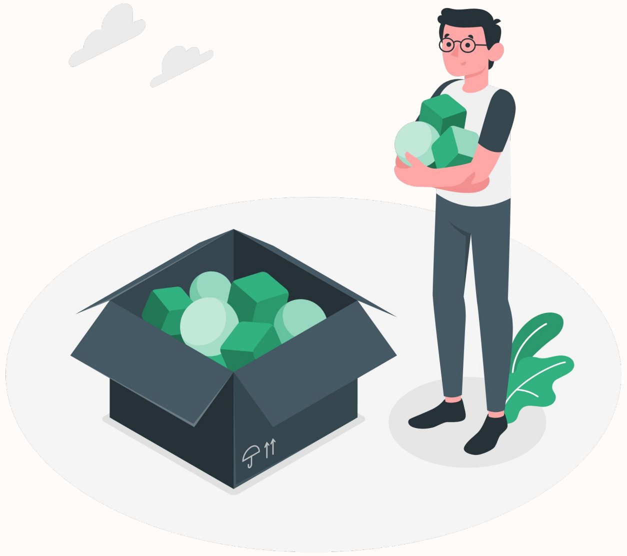
Job Connector Program

What is caching ?

Caching is the process of storing copies of files in a cache or a temporary storage location so that they can be accessed more quickly.

Why do we cache ?

- To save cost. Such as paying for bandwidth or even volume of data sent over the network.
- To reduce app response time.



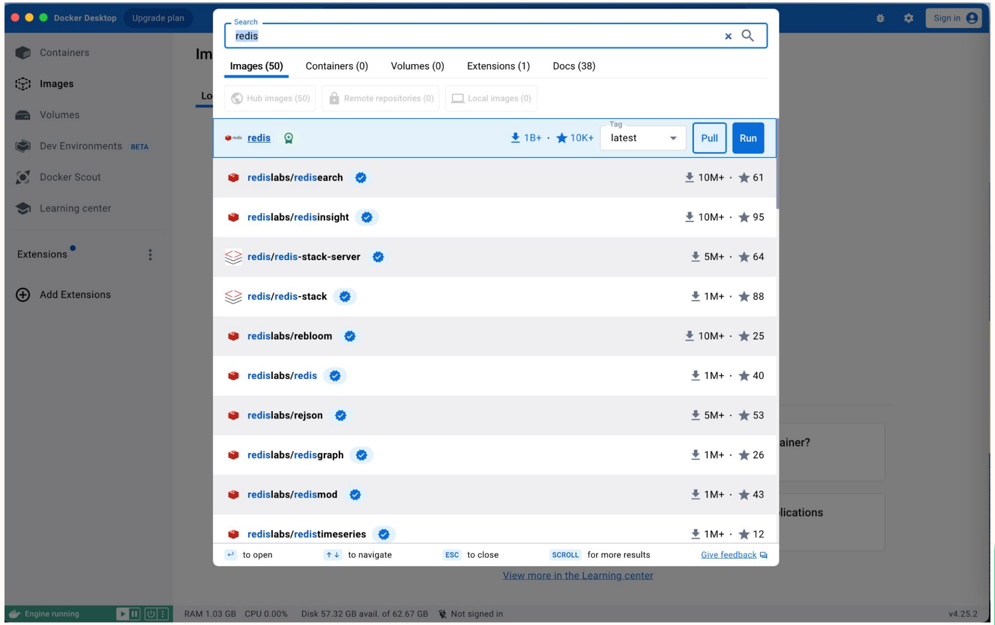
What is Redis ?

Redis Stands for **Remote Dictionary Server**. The open source, in-memory data store used by millions of developers as a database, cache, streaming engine, and message broker.

Setup Redis in Docker : Pull Redis Images

In this case, we will try to get an images from **docker desktop**

This time we will get redis images, open images menu and input “redis” in search bar. After that lets run docker images once more. See the difference!



redis

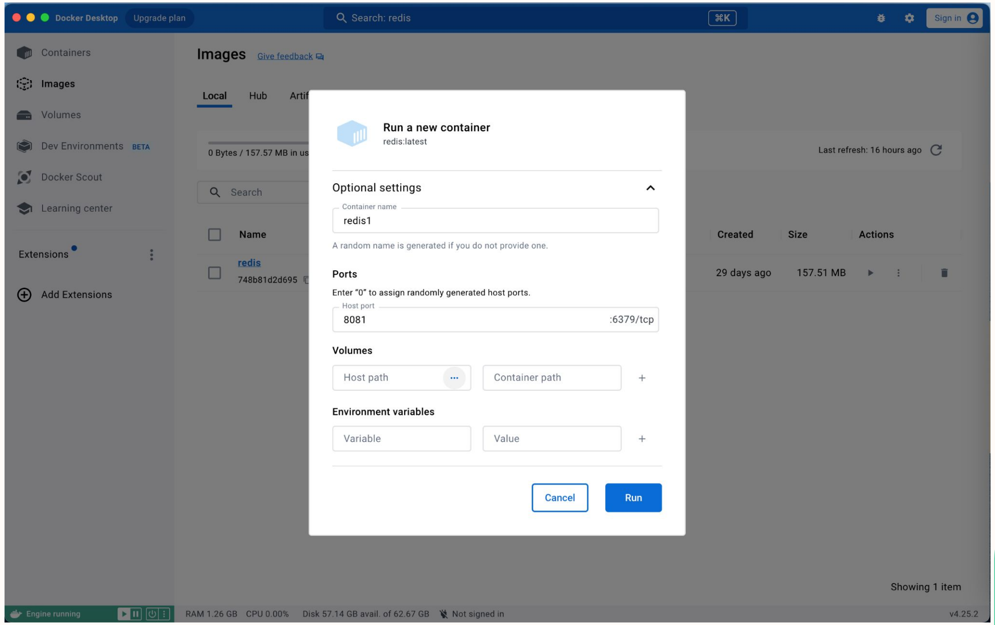
Image	Downloads	Stars
redis (Tag: latest)	1B+	10K+
redislabs/redisearch	10M+	61
redislabs/redisinsight	10M+	95
redis/redis-stack-server	5M+	64
redis/redis-stack	1M+	88
redislabs/rebloom	10M+	25
redislabs/redis	1M+	40
redislabs/rejson	5M+	53
redislabs/redisgraph	1M+	26
redislabs/redismod	1M+	43
redislabs/redistimeseries	1M+	12

RAM 1.03 GB CPU 0.00% Disk 57.32 GB avail. of 62.67 GB Not signed in

Run Redis Images

In this case, we will try to get an images from **docker desktop**

This time we will get redis images, open images menu and input “redis” in search bar. After that lets run docker images once more. See the difference!



Run a new container
redis:latest

Optional settings

Container name: redis1

A random name is generated if you do not provide one.

Ports

Enter “0” to assign randomly generated host ports.

Host port: 8081 :6379/tcp

Volumes

Host path | Container path

Environment variables

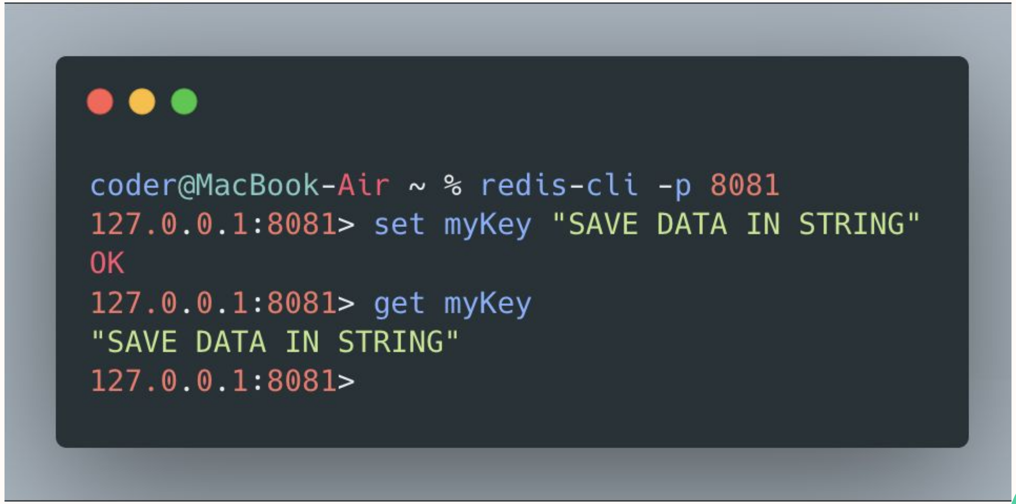
Variable | Value

Cancel | Run

Try to access redis from terminal

Lets test your redis from outside the container. Install redis-cli first. You can view redis documentation.

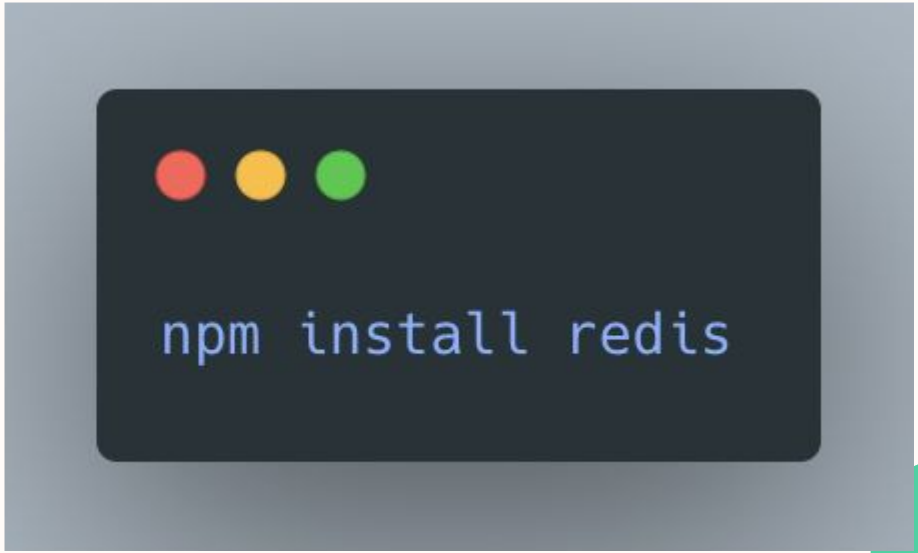
Try to run the redis. You just successfully accessing redis from docker container with custom port.



```
coder@MacBook-Air ~ % redis-cli -p 8081
127.0.0.1:8081> set myKey "SAVE DATA IN STRING"
OK
127.0.0.1:8081> get myKey
"SAVE DATA IN STRING"
127.0.0.1:8081>
```

Install redis in project

Don’t forget to install redis package into your projects.

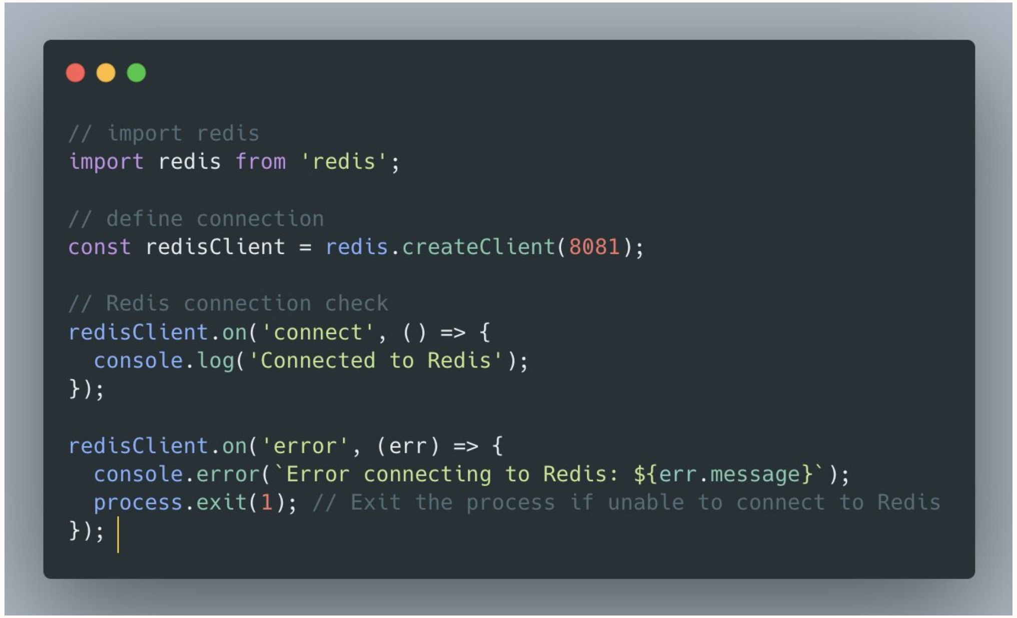


```
npm install redis
```

Implementation Redis in project

This is a setup for redis in your server app.

By default redis is using port 6379 and in this code we use **8081**.

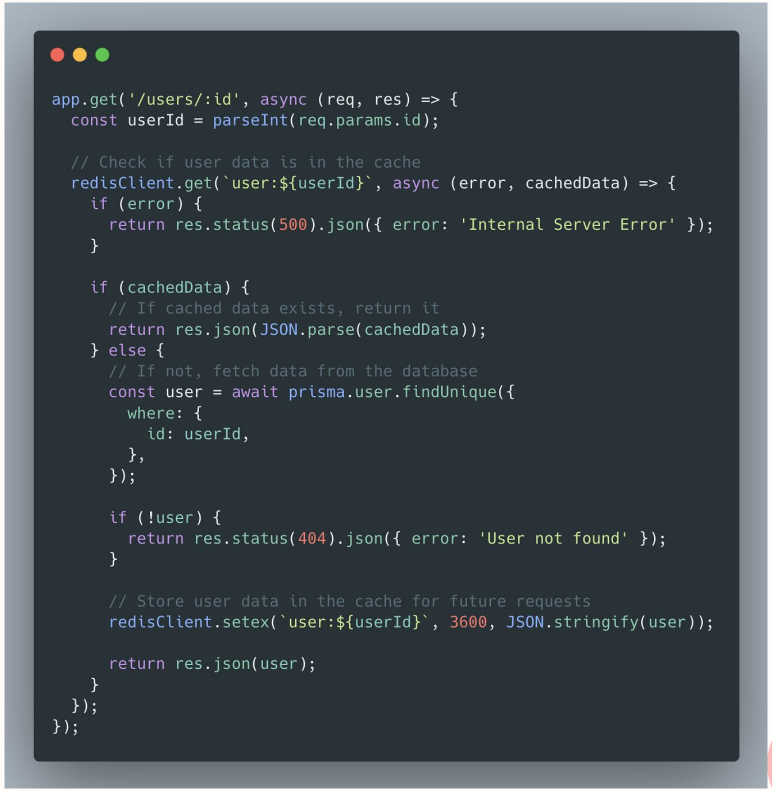


```
// import redis
import redis from 'redis';

// define connection
const redisClient = redis.createClient(8081);

// Redis connection check
redisClient.on('connect', () => {
  console.log('Connected to Redis');
});

redisClient.on('error', (err) => {
  console.error(`Error connecting to Redis: ${err.message}`);
  process.exit(1); // Exit the process if unable to connect to Redis
});
```

Implementation Redis in project

redisClient.get method is used to get data source from cache storage.

redisClient.setex method is used to keep and cache the data with expires time. Data would be keep in string type.



```
app.get('/users/:id', async (req, res) => {
  const userId = parseInt(req.params.id);

  // Check if user data is in the cache
  redisClient.get(`user:${userId}`, async (error, cachedData) => {
    if (error) {
      return res.status(500).json({ error: 'Internal Server Error' });
    }

    if (cachedData) {
      // If cached data exists, return it
      return res.json(JSON.parse(cachedData));
    } else {
      // If not, fetch data from the database
      const user = await prisma.user.findUnique({
        where: {
          id: userId,
        },
      });

      if (!user) {
        return res.status(404).json({ error: 'User not found' });
      }

      // Store user data in the cache for future requests
      redisClient.setex(`user:${userId}`, 3600, JSON.stringify(user));

      return res.json(user);
    }
  });
});
```

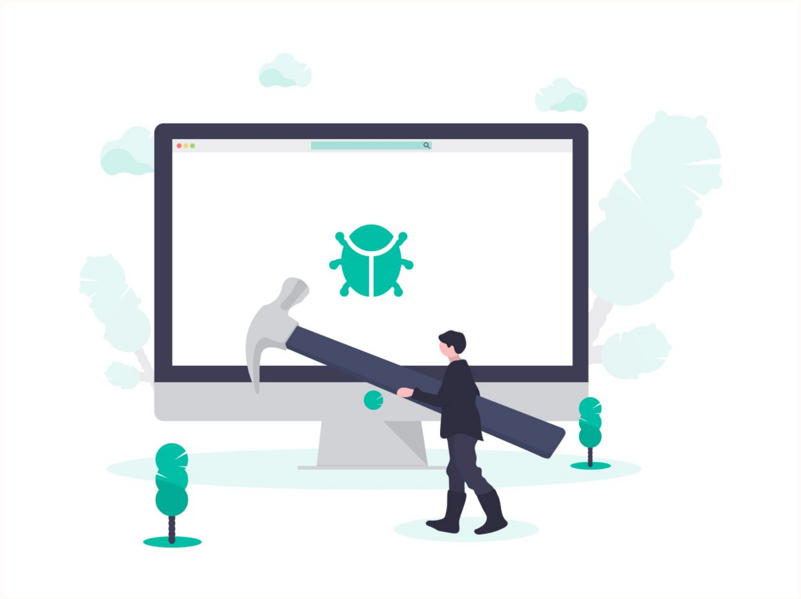
Logging

Logging are crucial for monitoring and troubleshooting your Node.js app. There are several ways to implement logging :

- Using runtimes console
- Using built in modules
- Using 3rd party library

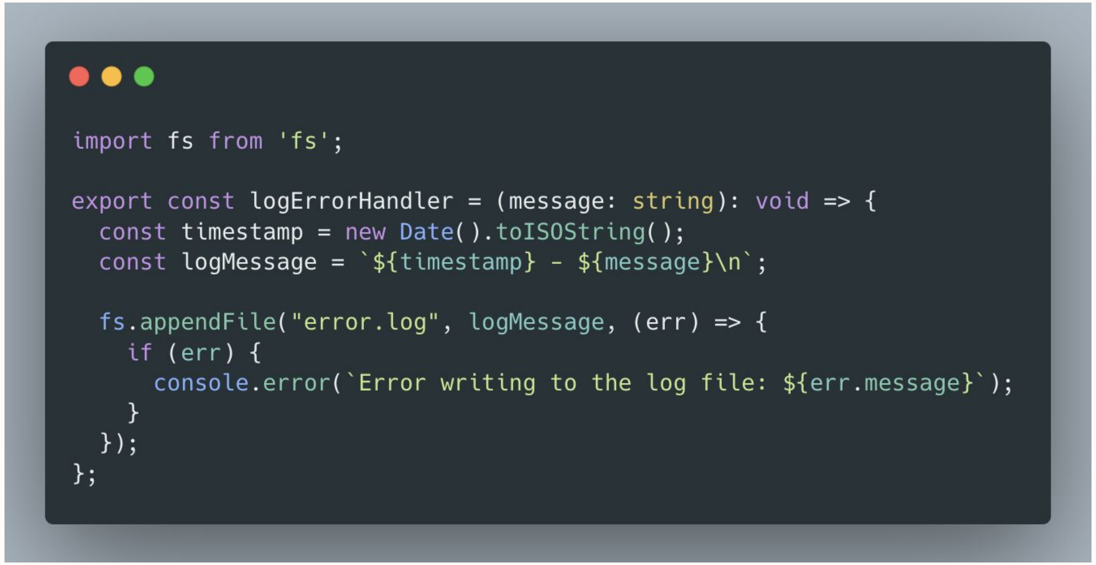
Using Runtimes Console

- console.log()
  - Purpose: Used for general information or logging.
- console.warn()
  - Purpose: Used for warning message.
- console.error()
  - Purpose: Used for error.



Using Built in Modules : **FS**

We can use fs to create log file. With log file we can store error message and capture error history. Create a **logErrorHandler** function and import in your middleware error.



```
import fs from 'fs';

export const logErrorHandler = (message: string): void => {
  const timestamp = new Date().toISOString();
  const logMessage = `${timestamp} - ${message}\n`;

  fs.appendFile("error.log", logMessage, (err) => {
    if (err) {
      console.error(`Error writing to the log file: ${err.message}`);
    }
  });
};
```

Using 3rd Party Library

There are several popular library to use for logging, for example Pino and Winston

Thank You!