bork bork!

Final Project ISFWS

Birth Rate Down, “Happy Dog” Rate Up

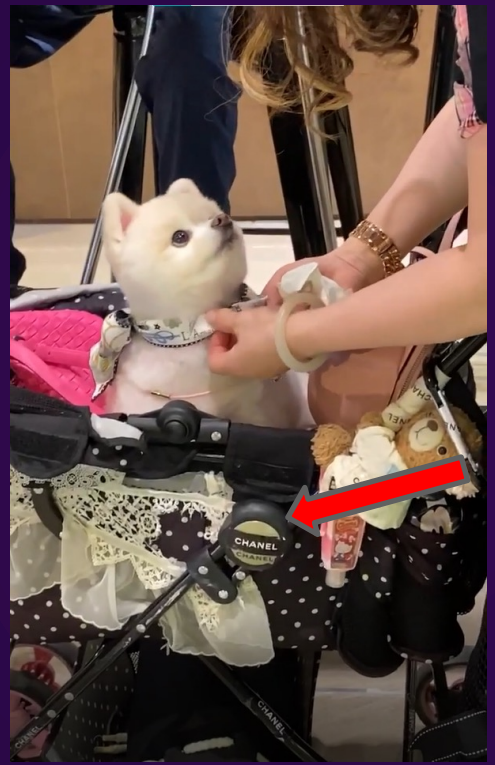
Professor: Alexander Yohan, Teacher Assistant: Jeremy Hartono

Mi5117701 Ford Iammongkol
M10815809 Eric Dao / 姚昭宇

1

Intro

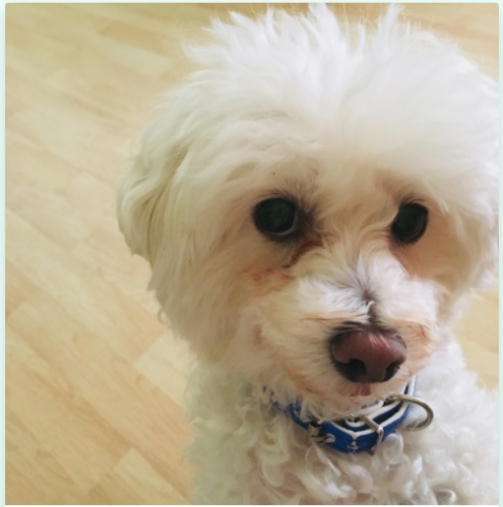
Dogs in Taiwan


CHANEL


CHANEL

Meet the Team

Ford
CS student servant of a corgi



Eric
CS student servant of a dog named Maxi

About bork bork

"bork bork" was made to both make enjoyment and realization of the many instances of dogs being treated like little children in Taiwan. It is both incredibly funny and entertaining. However, at the same time, a possible warning of a low birth-rate population choosing to raise dogs rather than children.

2

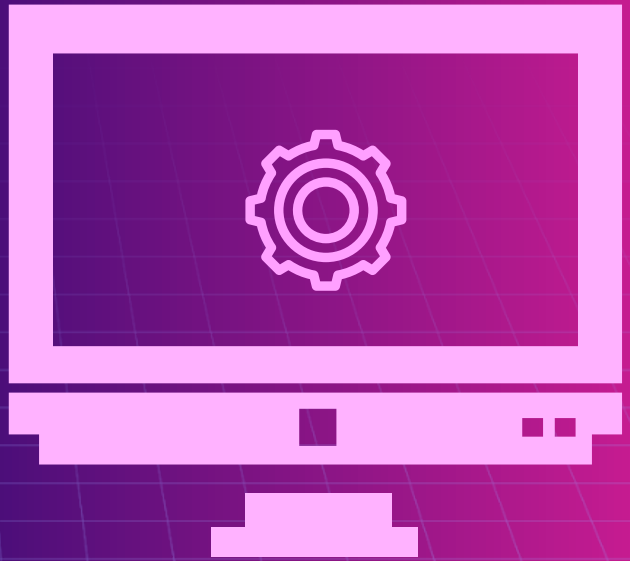
Our Setup

Our Setup

Operating system	**Windows 10**
Front-end	**html, css, js** Bootstrap 4
Back-end	**Nodejs v14.15.0** express, Passport, jsonwebtoken
Database	**Mysql v8.0**
Hosting	Hosting on an NTUST lab ip. A proxy heroku url is used to redirect to the lab ip.

Our Services

1. View the current posts
2. Create your own post
3. View Instagram posts



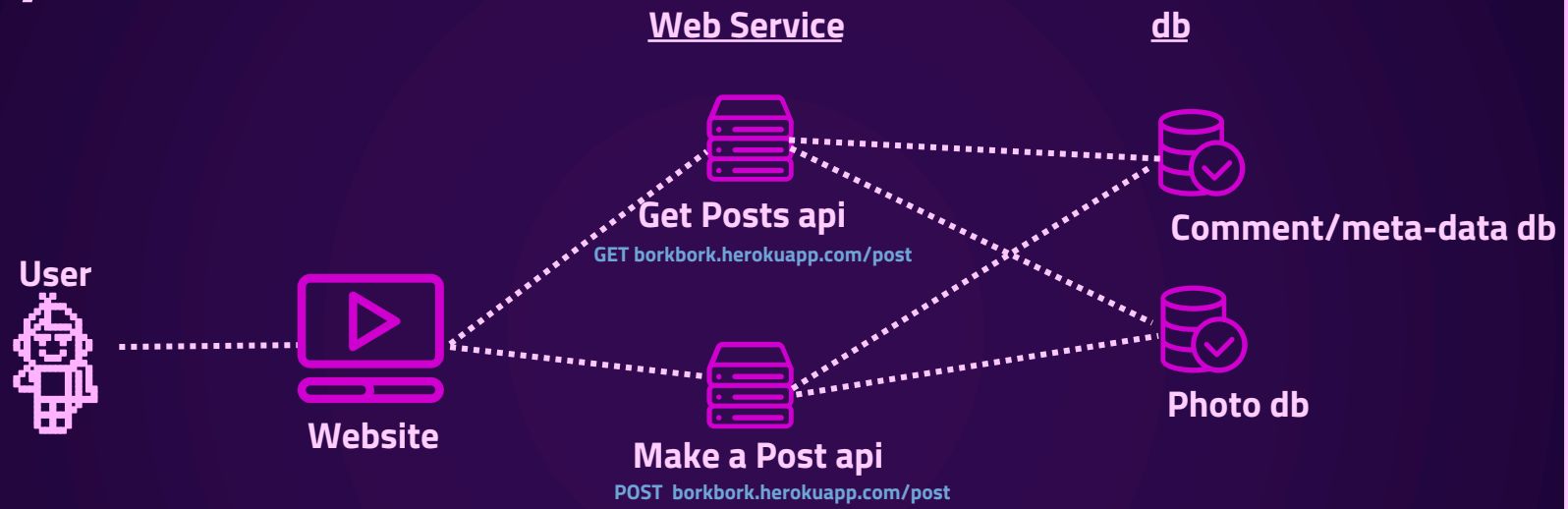
System Architecture

Web Service

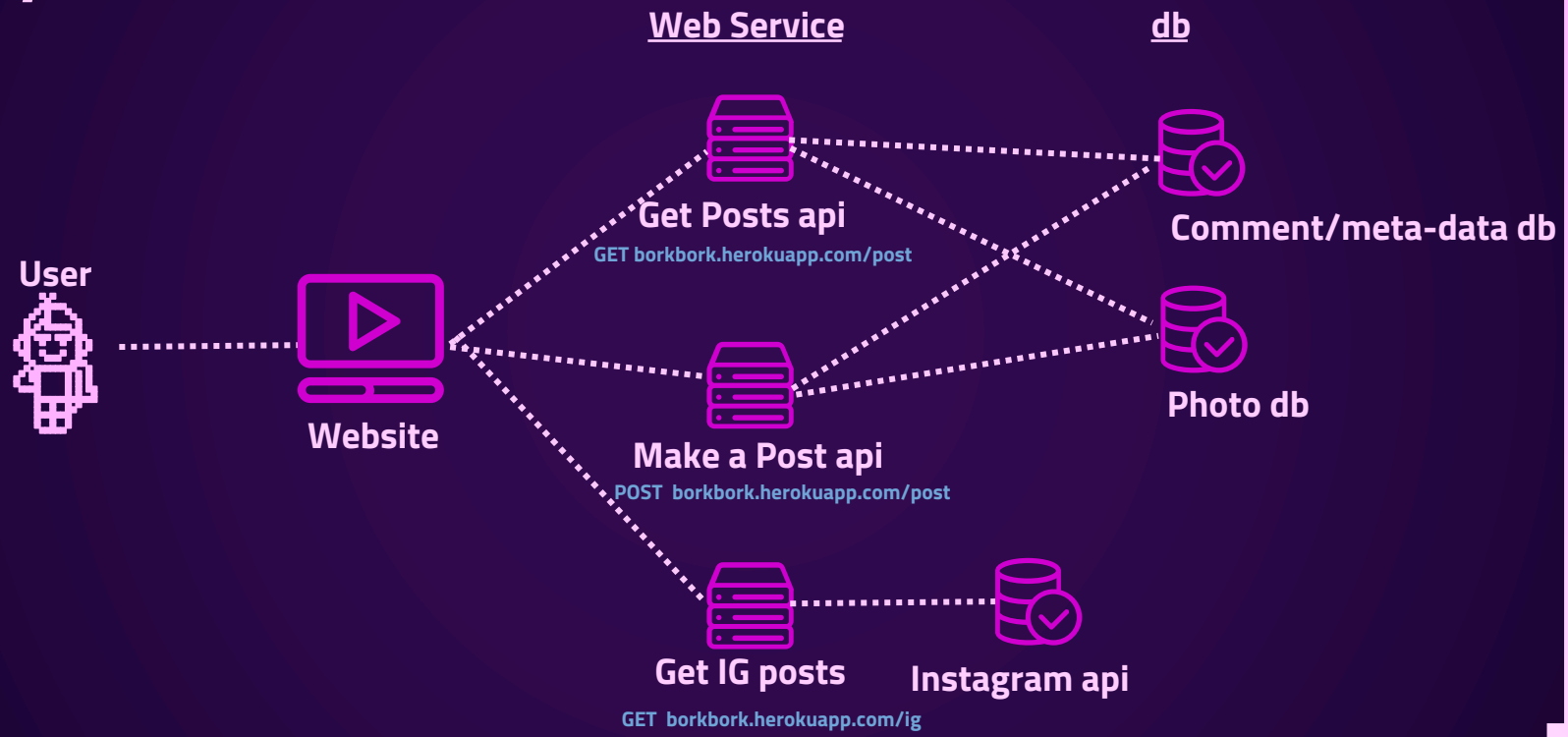
db

System Architecture

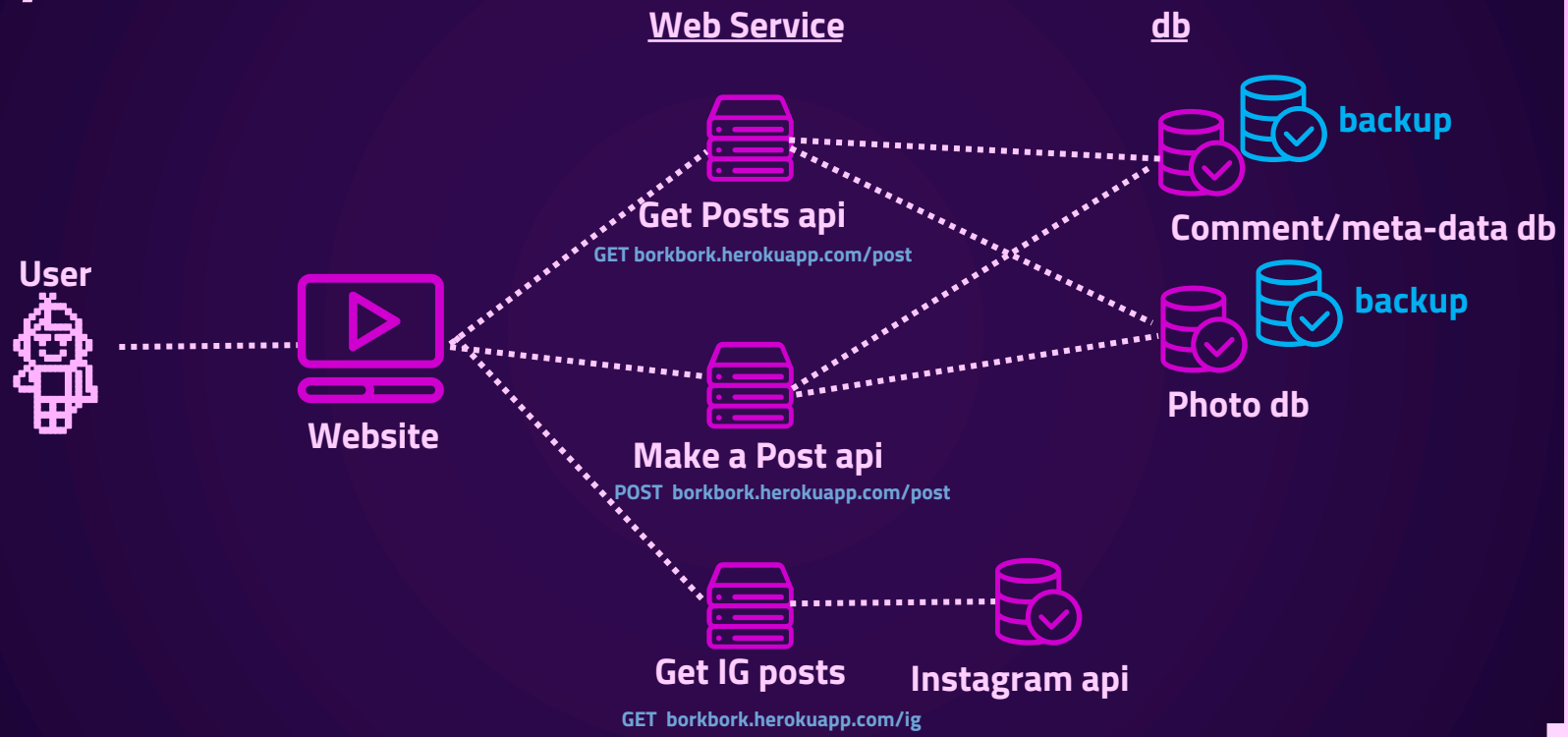
System Architecture



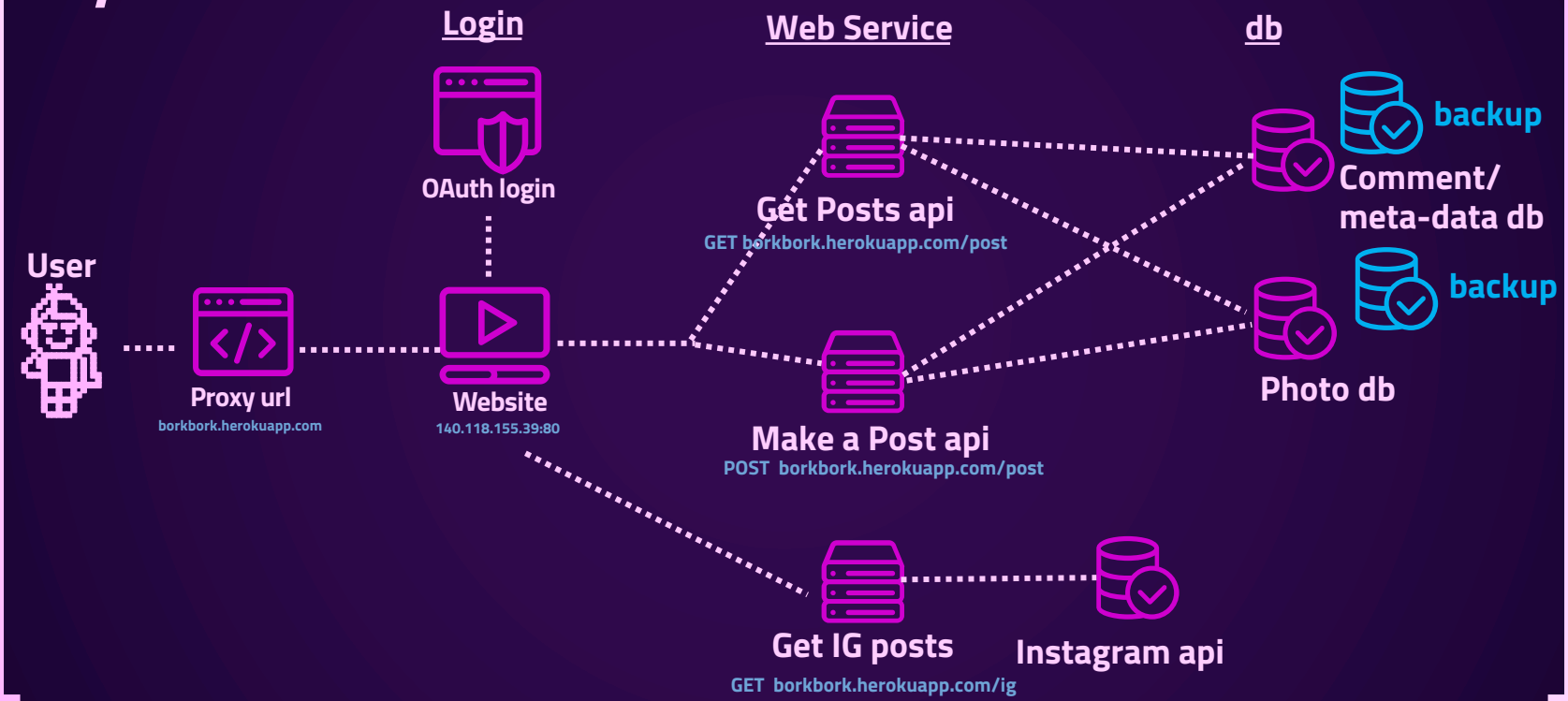
System Architecture



System Architecture



System Architecture

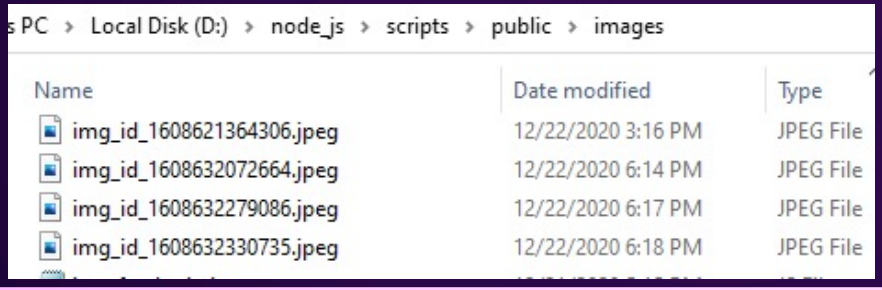


Databases

Comment/meta-data db

```
mysql> desc post;
+-----------+--------------+------+-----+-------------------+-------------------+
| Field     | Type         | Null | Key | Default           | Extra             |
+-----------+--------------+------+-----+-------------------+-------------------+
| post_id   | bigint       | NO   |     | NULL              |                   |
| post_text | varchar(500) | YES  |     | NULL              |                   |
| poster_id | bigint       | YES  |     | NULL              |                   |
| post_time | datetime     | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-----------+--------------+------+-----+-------------------+-------------------+
4 rows in set (0.00 sec)
```

Photo db



Accepted formats

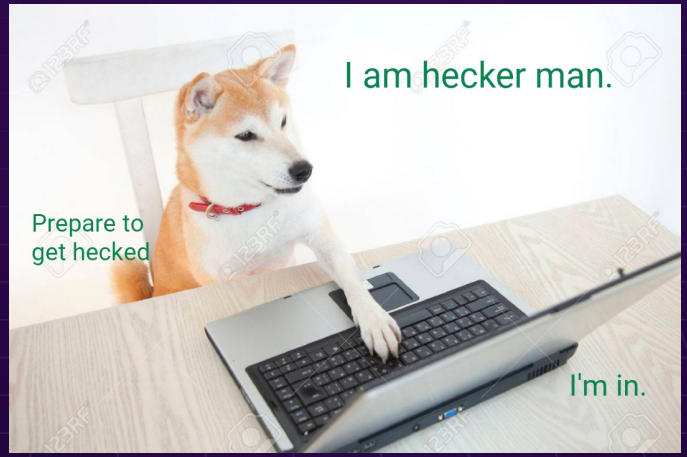
.jpg
.jpeg
.png
.gif

3

Demo Website

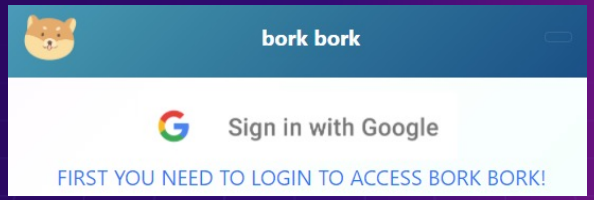
4

Web Service Security



Restricting access using OAuth

Restricting access using OAuth



Restricting access using OAuth

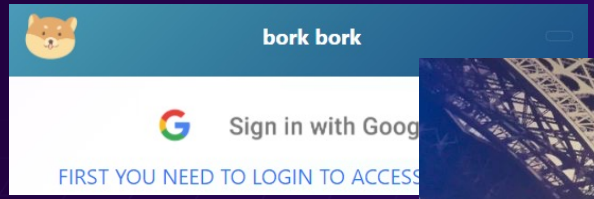
```
app.get('/auth/google', passport.authenticate('google', { scope: ['email', 'profile'] }), (req, res) => {
  console.log("Logging in via Google");
});

app.get('/auth/google/callback', passport.authenticate('google', { scope: ['email', 'profile'] }), (req, res) => {
  console.log(req.user);
  console.log('Callback landed');
  // Passportjs sends back the user attached to the request object, I set it as part of the session
  req.session.user = req.user;
  //sign jwt
  let accessToken = jwt.sign({ userId: req.user.id }, '524d4c5d7a2c71af42dfaf9094e93a029ffb8bfb1885444a986852dd62fc4d0a', {
    algorithm: "HS512",
    expiresIn: 86400
  });
  res.cookie("user", req.user.id, { secure: true, httpOnly: true, path: '/' })
  res.cookie("jwt", accessToken, { secure: true, httpOnly: true, path: '/' })
  // Redirect to budgeteer after the session has been set
  res.redirect("/");
});
```

Restricting access using OAuth

D: > node_js > scripts > middleware.js > verify > verify

```
1   //const jwt = require('json-web-token')
2   const jwt = require('jsonwebtoken');
3
4   exports.verify = function(req, res, next){
5       let accessToken = req.cookies.jwt
6
7       //if there is no token stored in cookies, the request is redirected to login page
8       if (!accessToken){
9           //return res.status(403).send()
10          return res.redirect('/login');
11      }
12
13      let payload
14      try{
15          //use the jwt.verify method to verify the access token
16          //throws an error if the token has expired or has a invalid signature
17          console.log(accessToken);
18          console.log('524d4c5d7a2c71af42dfaf9094e93a029ffb8bfb1885444a986852dd62fc4d0a');
19          payload = jwt.verify(accessToken, '524d4c5d7a2c71af42dfaf9094e93a029ffb8bfb1885444a98
20          console.log(payload);
21          next()
22      }
23      catch(e){
24          //if an error occured return request unauthorized error
25          console.log(e)
26          res.clearCookie("jwt");
27          if(e instanceof jwt.TokenExpiredError) {
28              return res.redirect('/login');
29          }
30          return res.status(401).send()
```

Restricting access using OAuth



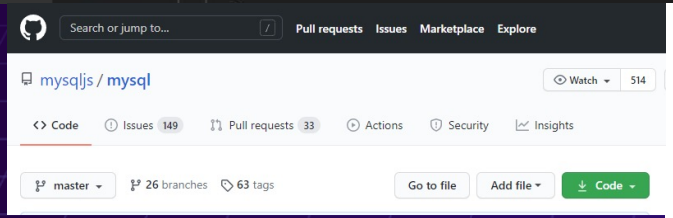

```
app.get('/auth/google', passport.authenticate('google',
  console.log("Logging in via Google");
});

app.get('/auth/google/callback', passport.authenticate(
  console.log(req.user);
  console.log('Callback landed');
  // Passportjs sends back the user attached to the req
  req.session.user = req.user;
  //sign jwt
  let accessToken = jwt.sign({ userId: req.user.id },
    algorithm: "HS512",
    expiresIn: 86400
  });
  res.cookie("user", req.user.id, { secure: true, httpO
  res.cookie("jwt", accessToken, { secure: true, httpOn
  // Redirect to budgeteer after the session has been s
  res.redirect("/");
});
```

```
verify
n')

next){
rt

cookies, the request is unauthorized

)

to verify the access token
en has expired or has a invalid signature

f42dfaf9094e93a029ffb8bfb1885444a986852dd62fc4d0a');
oken, '524d4c5d7a2c71af42dfaf9094e93a029ffb8bfb1885444a986852dd62fc4d0a', { algorithms: ['HS512'] })

request unauthorized error

iredError) {
gin');
        }
        return res.status(401).send()
    }
}
```

Sql injection risk

Sql injection risk

```
Post.findById = (postId, result) => {
  //const escaper = sql.escape(postId);
  sql.query(`SELECT * FROM post WHERE post_id = ${postId}`, (err, res) => {
    if (err) {
      console.log("error: ", err);
      result(err, null);
      return;
    }
```



Different value types are escaped differently, here is how:

- Numbers are left untouched
- Booleans are converted to `true` / `false`
- Date objects are converted to `'YYYY-mm-dd HH:ii:ss'` strings
- Buffers are converted to hex strings, e.g. `X'0fa5'`
- Strings are safely escaped
- Arrays are turned into list, e.g. `['a', 'b']` turns into `'a', 'b'`
- Nested arrays are turned into grouped lists (for bulk inserts), e.g. `[['a', 'b'], ['c', 'd']]` turns into `('a', 'b'), ('c', 'd')`
- Objects that have a `toSqlString` method will have `.toSqlString()` called and the returned value is used as the raw SQL.
- Objects are turned into `key = 'val'` pairs for each enumerable property on the object. If the property's value is a function, it is skipped; if the property's value is an object, toString() is called on it and the returned value is used.
- `undefined` / `null` are converted to `NULL`
- `NaN` / `Infinity` are left as-is. MySQL does not support these, and trying to insert them as values will trigger MySQL errors until they implement support.

Home Workspaces Reports Explore Search Postman Invite Upgrade

My Workspace New Import

Collections APIs Environments Mock Servers Monitors History

test

GET http://localhost:300...

No Environment

a / http://localhost:3000/post

Save

GET http://localhost:3000/post/1608621364006 OR 1=1; Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON


```
[
    {
        "post_id": 1608460935042,
        "post_text": "This is a sample post to test everything out.",
        "poster_id": 0,
        "liked_count": 0,
        "post_time": "2020-12-20T10:42:15.000Z"
    },
    {
        "post_id": 1608464611544,
        "post_text": "Another sample post to test things out",
        "poster_id": 0,
        "liked_count": 0,
        "post_time": "2020-12-20T11:43:31.000Z"
    },
    {
        "post_id": 1608555474457,
        "post_text": "Testing everything once again after having made some changes.",
        "poster_id": 0,
        "liked_count": 0,
        "post_time": "2020-12-21T12:57:54.000Z"
    },
    {
        "post_id": 1608616576754,
        "post_text": "Trying once again another post",
        "poster_id": 0,
        "liked_count": 0,
```


```
D:\node_js\scripts>node server.js
Server is running on port 3000.
Successfully connected to the database.
found post:  1 OR 1=1; [
  RowDataPacket {
    post_id: 1608460935042,
    post_text: 'This is a sample post to test everything out.
    poster_id: 0,
    liked_count: 0,
    post_time: 2020-12-20T10:42:15.000Z
  },
  RowDataPacket {
    post_id: 1608464611544,
    post_text: 'Another sample post to test things out',
    poster_id: 0,
    liked_count: 0,
    post_time: 2020-12-20T11:43:31.000Z
  },
  RowDataPacket {
    post_id: 1608555474457,
    post_text: 'Testing everything once again after having ma
    poster_id: 0,
    liked_count: 0,
    post_time: 2020-12-21T12:57:54.000Z
```

Console Desktop Agent Runner Trash

6:02 PM 12/22/2020

Sql injection risk

```
Post.findById = (postId, result) => {
  const escaper = sql.escape(postId);
  sql.query(`SELECT * FROM post WHERE post_id = ${escaper}`,
```

```
D:\node_js\scripts>node server.js
Server is running on port 3000.
Successfully connected to the database.
1324; OR 1=1;
'1324; OR 1=1;'
```

a / http://localhost:3000/post

GET http://localhost:3000/post/1 OR 1=1;

Params Authorization Headers (9) Body Pre-request Scrip

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "message": "Not found Post with id 1 OR 1=1;."
3 }
```

Sql injection risk

Using “?” characters as placeholders for values to be escaped

```
Post {
  post_id: 1609052950543,
  post_text: 'kldasjlkjfaa; DELETE FROM post WHERE post_id = 1; ',
  poster_id: 0
}
```

```
3   // constructor
4   const Post = function(post) {
5     this.post_id = post.post_id;
6     this.post_text = post.post_text;
7     this.poster_id = post.poster_id;
8   };
9
10
11  Post.create = (newPost, result) => {
12    sql.query(`INSERT INTO post SET ?`, newPost, (err, res) => {
13      if (err) {
14        console.log("error: ", err);
15        result(err, null);
16        return;
17      }
```

Sql injection risk

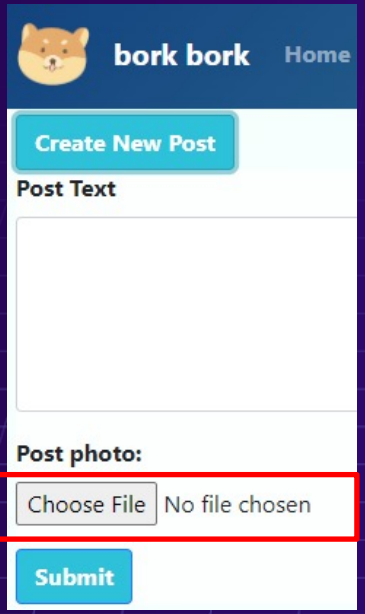
Using “?” characters as placeholders for values to be escaped

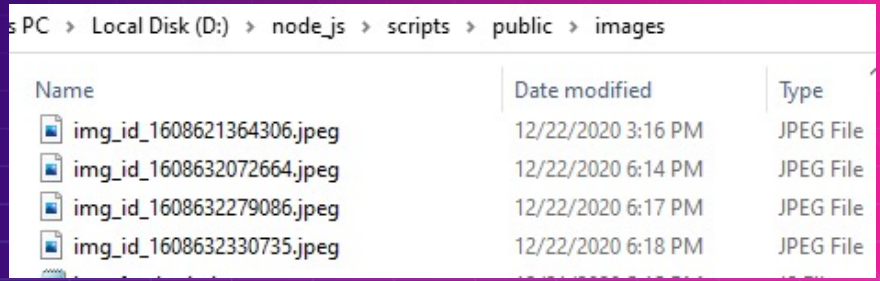
```
3   // constructor
4   const Post = funct
5     this.post_id = p
6     this.post_text =
7     this.poster_id =
8   };
9
10
11  Post.create = (nev
12    sql.query(`INSEF
13      if (err) {
14        console.log(
15        result(err,
16        return;
17      }
```

```
DELETE FROM post WHERE post_id = 1; ',
```

Malicious file upload risk

Malicious file upload risk



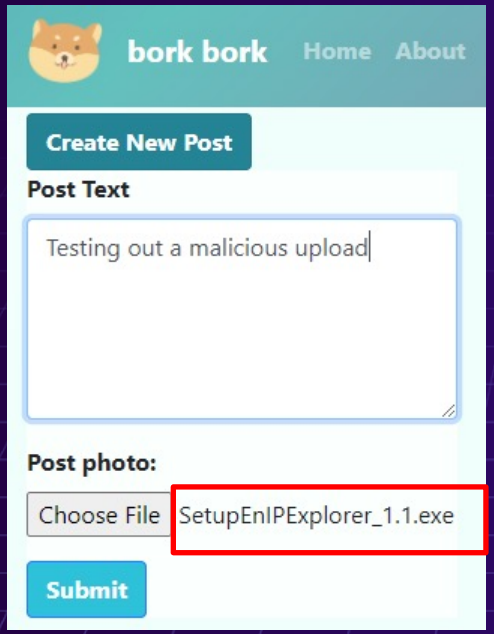


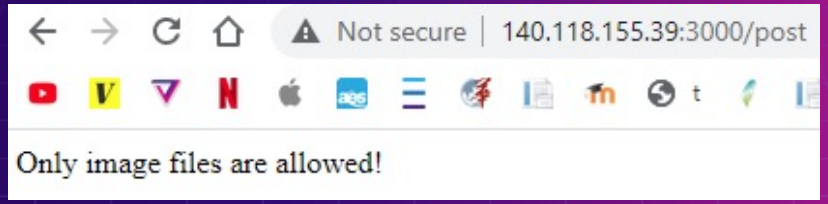
Malicious file upload risk

D: > node_js > scripts > app > controllers > JS helpers.js > ...

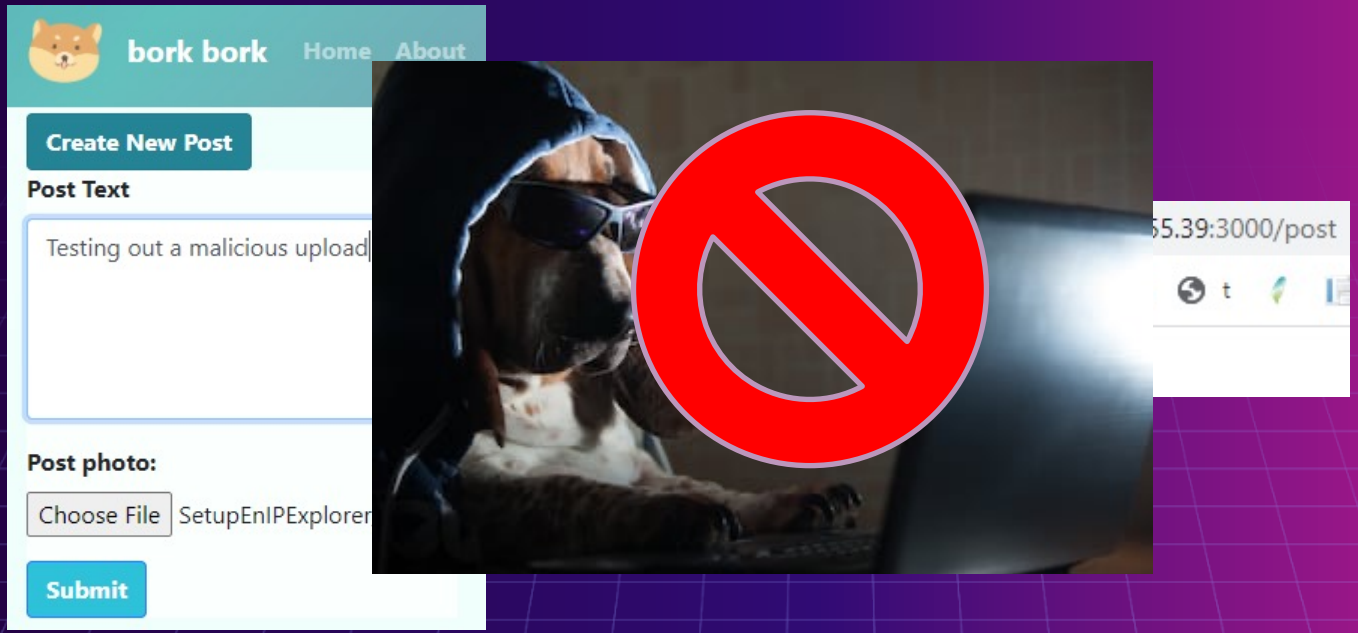
```
1   const imageFilter = function(req, file, cb) {
2       // Accept images only
3       if (!file.originalname.match(/\.(jpg|JPG|jpeg|JPEG|png|PNG|gif|GIF)$/)) {
4           req.fileValidationError = 'Only image files are allowed!';
5           return cb(new Error('Only image files are allowed!'), false);
6       }
```

Malicious file upload risk



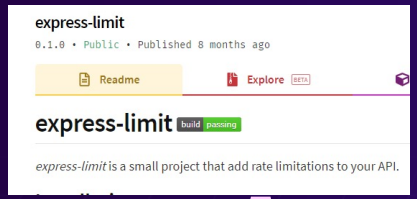


Malicious file upload risk



ddos risk /api abuse risk

ddos risk /api abuse risk



Accessing pages

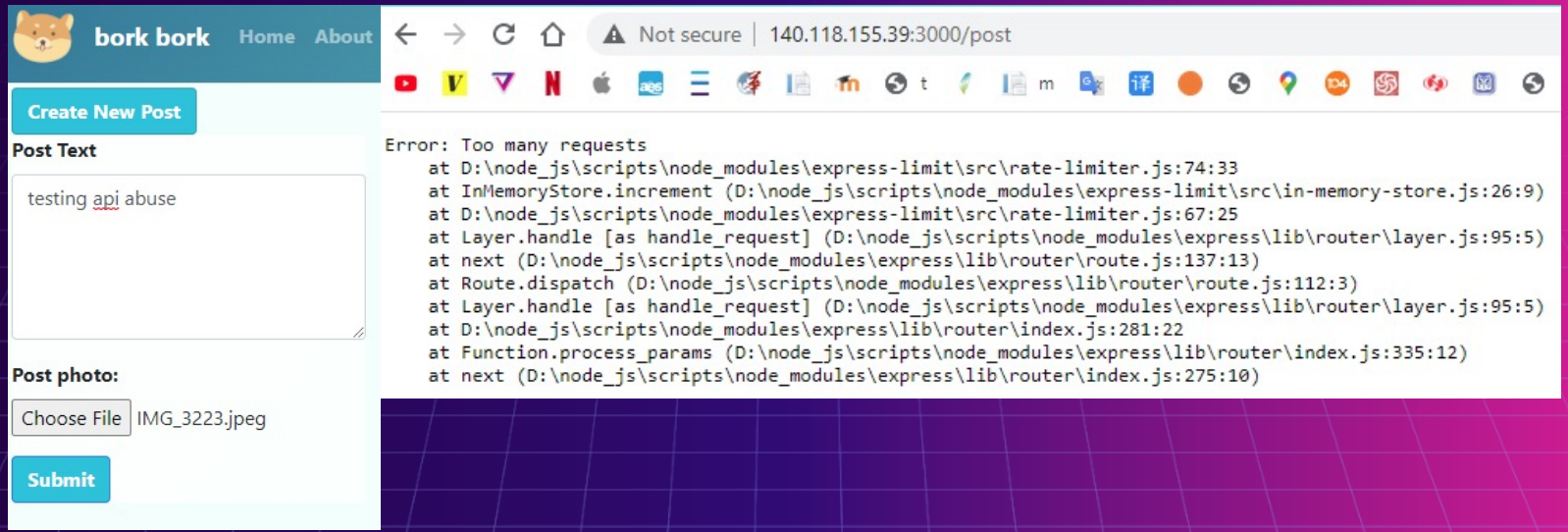
```
app.get("/", limit({
  max:    20,         // 20 requests
  period: 60 * 1000 // per minute (60 seconds)
}), (req, res) => {
  res.sendFile(path.join(__dirname, 'app/views/index.html'));
});
app.get("/about", limit({
  max:    20,
  period: 60 * 1000
}), (req, res) => {
  res.sendFile(path.join(__dirname, 'app/views/about.html'));
});
app.get("/contact", limit({
  max:    20,
  period: 60 * 1000
}), (req, res) => {
  res.sendFile(path.join(__dirname, 'app/views/contact.html'));
});
```

Accessing api

```
app.post('/post', limit({
  max:    5,         // 5 requests
  period: 60 * 1000 // per minute (60 seconds)
}), post.create);
//app.post("/post", post.create);
app.get("/post", limit({
  max:    20,
  period: 60 * 1000
}), post.findAll);
app.get("/post/:postId", limit({
  max:    20,
  period: 60 * 1000
}), post.findOne);
```

ddos risk /api abuse risk

Trying to exceed limit:
post over 5 times within 60 sec



ddos risk /api abuse risk

Trying to post
over 5 times within 60 sec

bork bork Home

Create New Post

Post Text

testing ddos

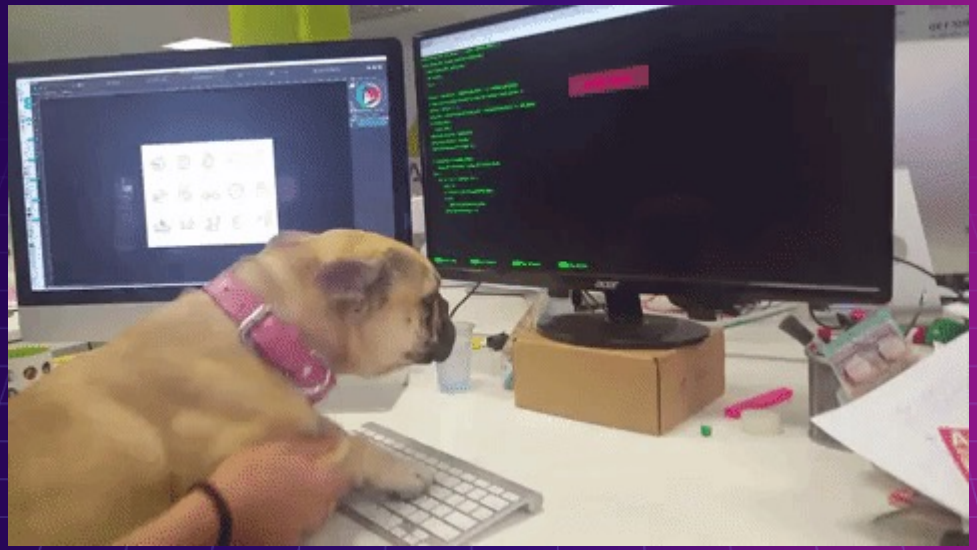
Post photo:

Choose File IMG_0833.jpeg

Submit

```
74:33
imit\src\in-memory-store.js:26:9)
67:25
express\lib\router\layer.js:95:5)
137:13)
\route.js:112:3)
express\lib\router\layer.js:95:5)
ib\router\index.js:335:12)
275:10)
```

Databases takedown risk



Sql backup database

Sql procedure

```
mysql> show procedure status like 'backup'
    -> ;
+-------+--------+-----------+----------------+---------------------+---------------------+---------------+
| Db    | Name   | Type      | Definer        | Modified            | Created             | Security_type |
+-------+--------+-----------+----------------+---------------------+---------------------+---------------+
| doggo | backup | PROCEDURE | root@localhost | 2020-12-29 21:46:49 | 2020-12-29 21:46:49 | DEFINER       |
+-------+--------+-----------+----------------+---------------------+---------------------+---------------+
1 row in set (0.31 sec)
```

Sql procedure code

```
DELIMITER //
CREATE PROCEDURE backup()
BEGIN
DROP TABLE IF EXISTS `post_backup`;
CREATE TABLE post_backup SELECT * FROM post;
END //
DELIMITER ;

DELIMITER //
CREATE EVENT mybackup
ON SCHEDULE EVERY 1 DAY
DO
  CALL backup(); //
DELIMITER ;
```

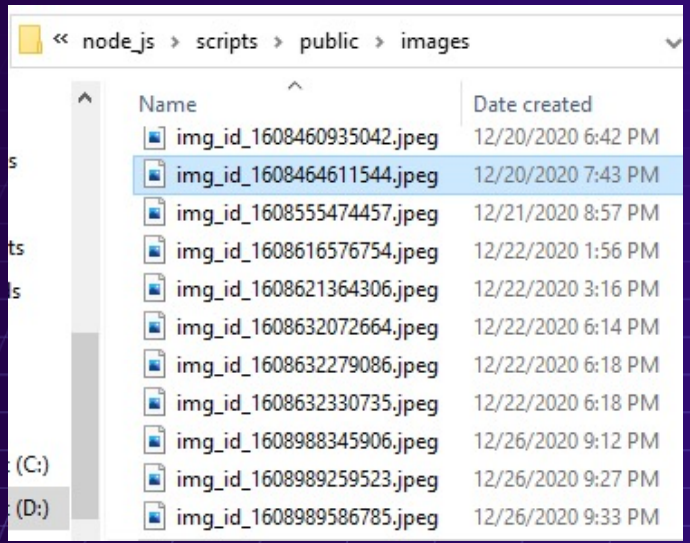
Database

```
mysql> select* from post;
+---------------+-----------------------------------------------------------+-----------+------------+---------------------+
| post_id       | post_text                                                 | poster_id | liked_count | post_time           |
+---------------+-----------------------------------------------------------+-----------+------------+---------------------+
| 1608460935042 | This is a sample post to test everything out.             |         0 |          0 | 2020-12-20 18:42:15 |
| 1608464611544 | Another sample post to test things out                    |         0 |          0 | 2020-12-20 19:43:31 |
| 1608555474457 | Testing everything once again after having made some changes. |         0 |          0 | 2020-12-21 20:57:54 |
| 1609616576754 | Trying once again another post                            |         0 |          0 | 2020-12-22 18:56:16 |
+---------------+-----------------------------------------------------------+-----------+------------+---------------------+
```

Database backup

```
mysql> select* from post_backup;
+---------------+-----------------------------------------------------------+-----------+------------+---------------------+
| post_id       | post_text                                                 | poster_id | liked_count | post_time           |
+---------------+-----------------------------------------------------------+-----------+------------+---------------------+
| 1608460935042 | This is a sample post to test everything out.             |         0 |          0 | 2020-12-20 18:42:15 |
| 1608464611544 | Another sample post to test things out                    |         0 |          0 | 2020-12-20 19:43:31 |
| 1608555474457 | Testing everything once again after having made some changes. |         0 |          0 | 2020-12-21 20:57:54 |
| 1609616576754 | Trying once again another post                            |         0 |          0 | 2020-12-22 18:56:16 |
+---------------+-----------------------------------------------------------+-----------+------------+---------------------+
```

image backup database



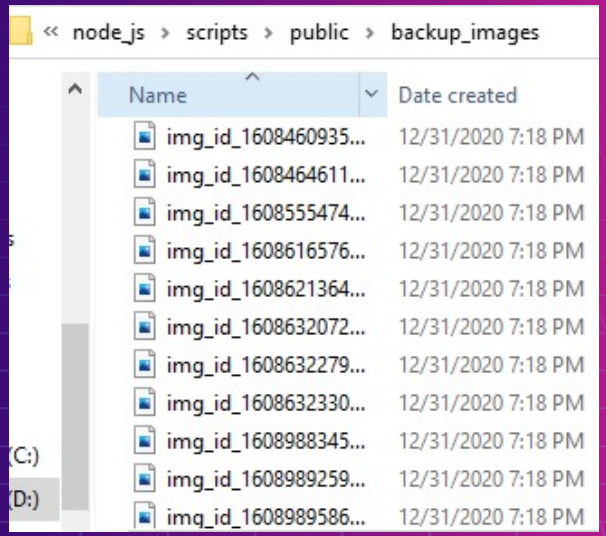
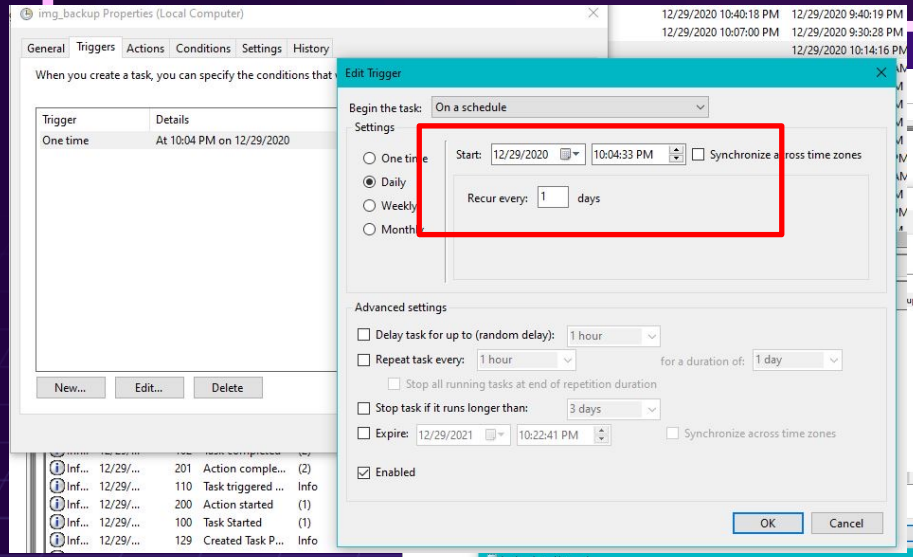
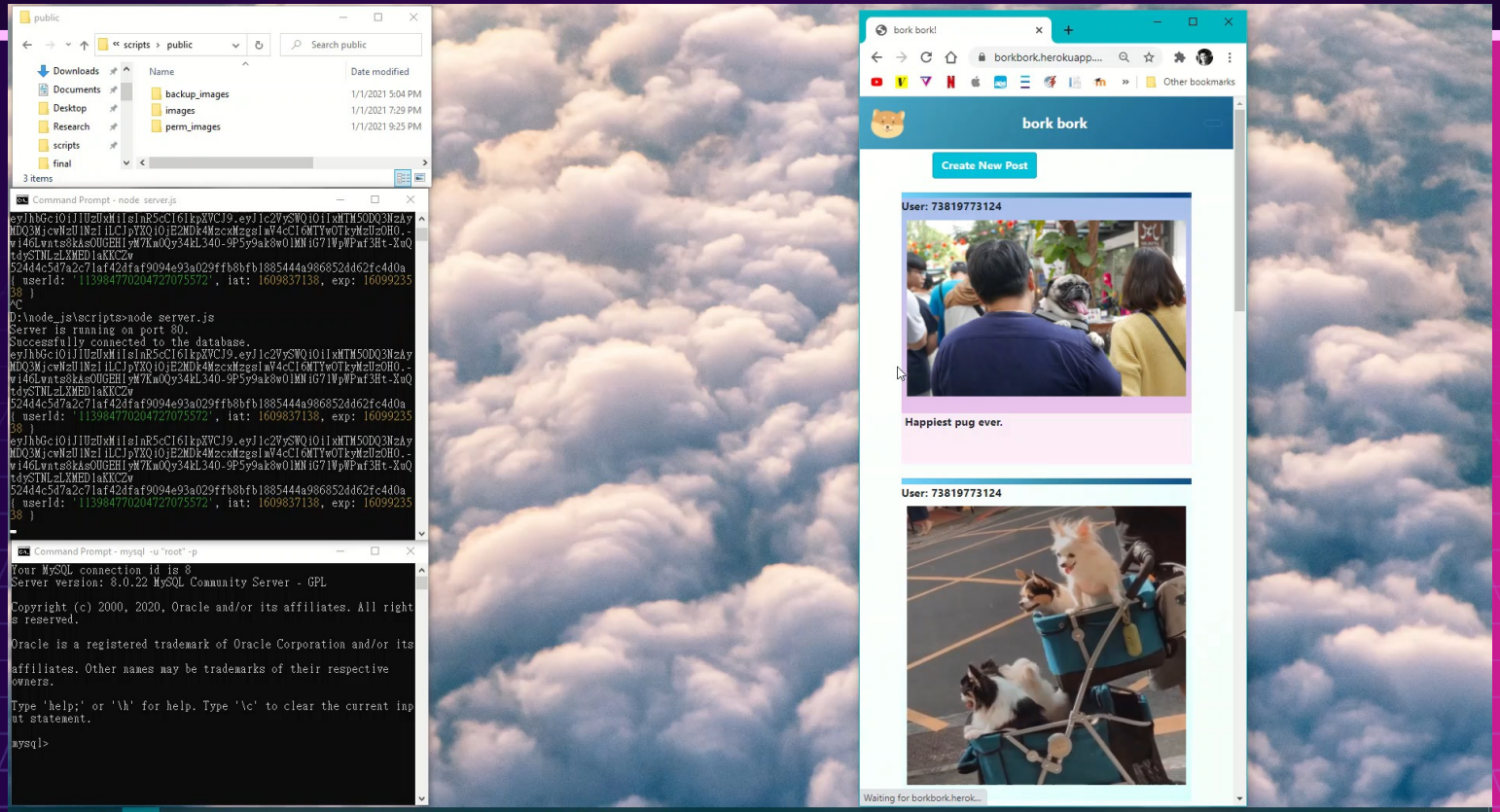


image backup database

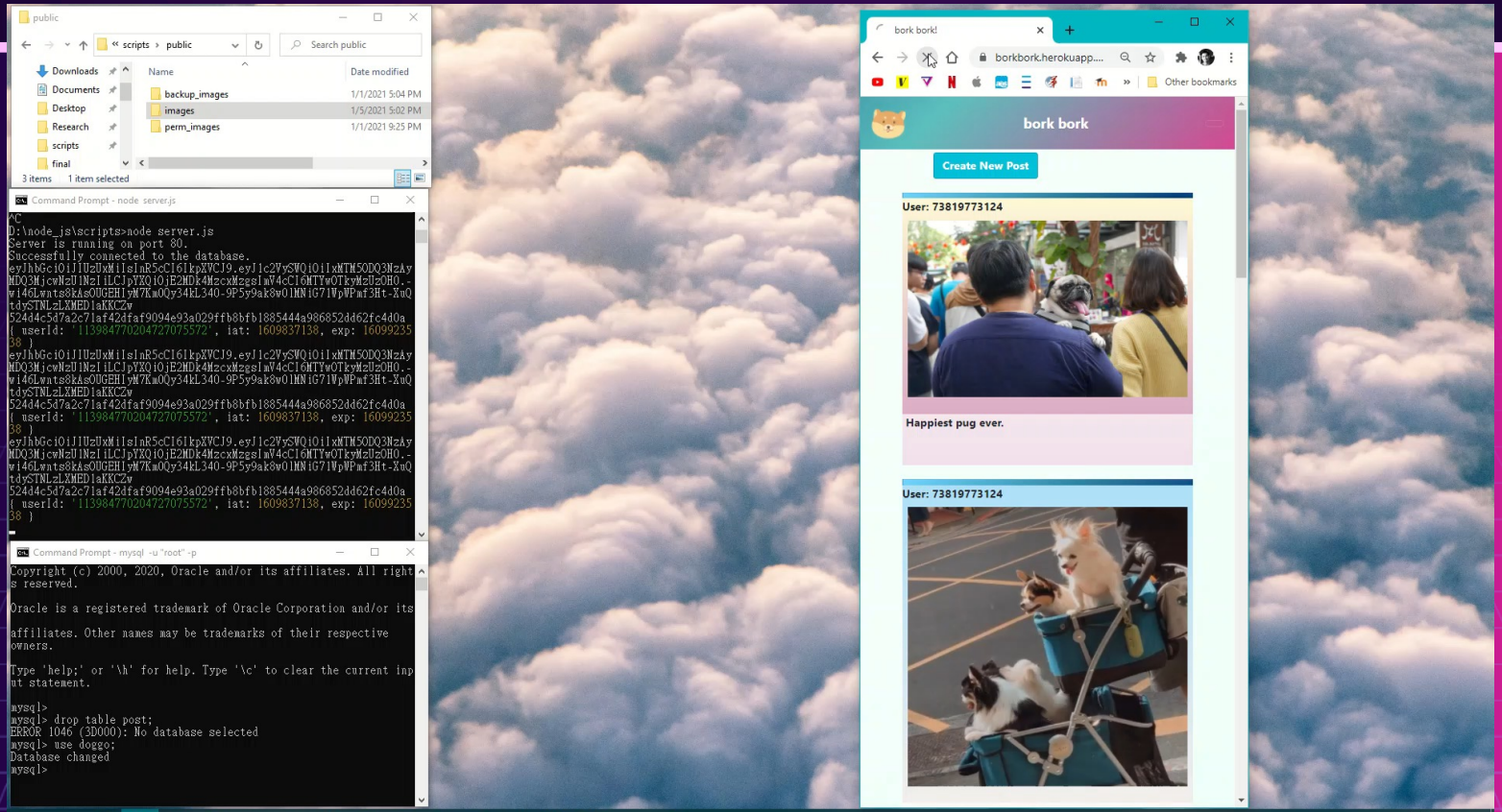


Demo backup databases

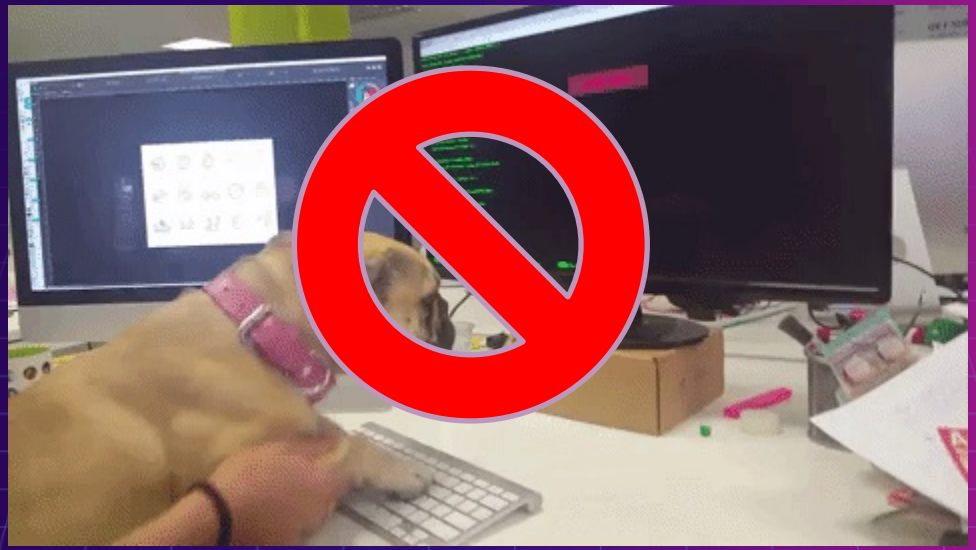
Image Database and Web Service takedown risk



SQL Database and Web Service takedown risk



Database takedown risk



More threats looming every corner

Thank you!

Any Questions?