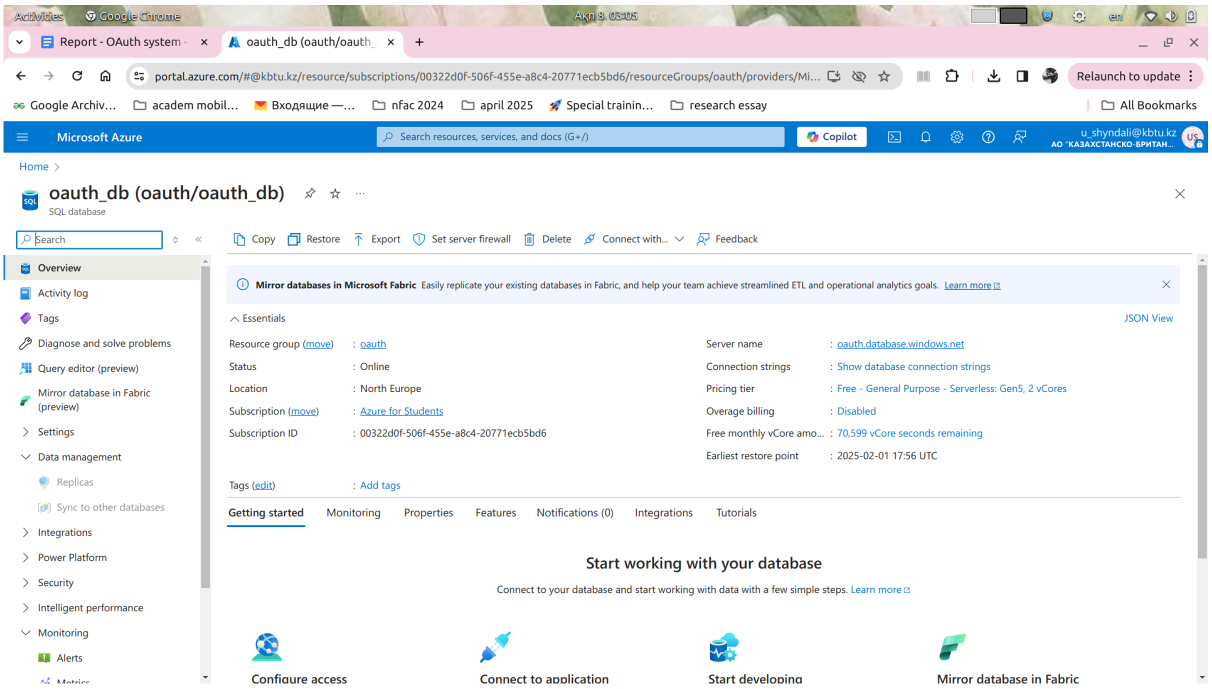
Report - OAuth system

Azure SQL was used:



Activities Google Chrome Акп 8 03:05

Report - OAuth system - × oauth_db (oauth/oauth_ ×

portal.azure.com/#@kbtu.kz/resource/subscriptions/00322d0f-506f-455e-a8c4-20771ecb5bd6/resourceGroups/oauth/providers/Mi... Relaunch to update

Google Archiv... academ mobil... Входящие —... nfac 2024 april 2025 Special trainin... research essay All Bookmarks

Microsoft Azure Search resources, services, and docs (G+/) Copilot u_shyndali@kbtu.kz АО "КАЗАХСТАНСКО-БРИТАН...

Home >

oauth_db (oauth/oauth_db)
SQL database

Copy Restore Export Set server firewall Delete Connect with... Feedback

Overview
Activity log
Tags
Diagnose and solve problems
Query editor (preview)
Mirror database in Fabric (preview)
Settings
Data management
Replicas
Sync to other databases
Integrations
Power Platform
Security
Intelligent performance
Monitoring
Alerts
Metrics

Mirror databases in Microsoft Fabric Easily replicate your existing databases in Fabric, and help your team achieve streamlined ETL and operational analytics goals. Learn more

Essentials JSON View

Resource group (move)	: oauth	Server name	: oauth.database.windows.net
Status	: Online	Connection strings	: Show database connection strings
Location	: North Europe	Pricing tier	: Free - General Purpose - Serverless: Gen5, 2 vCores
Subscription (move)	: Azure for Students	Overage billing	: Disabled
Subscription ID	: 00322d0f-506f-455e-a8c4-20771ecb5bd6	Free monthly vCore amo...	: 70,599 vCore seconds remaining
		Earliest restore point	: 2025-02-01 17:56 UTC
Tags (edit)	: Add tags		

Getting started Monitoring Properties Features Notifications (0) Integrations Tutorials

Start working with your database

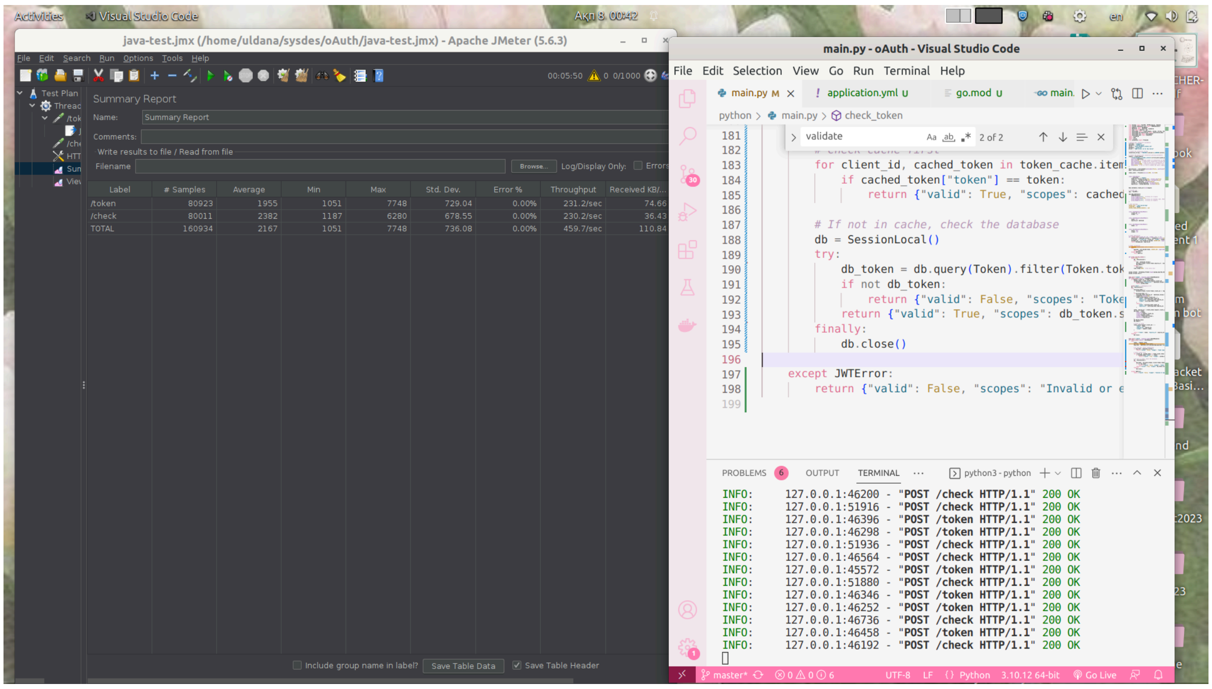
Connect to your database and start working with data with a few simple steps. Learn more

Configure access Connect to application Start developing Mirror database in Fabric

Nginx setup:

```
worker_processes auto;

events {
    worker_connections 1024;
}
http {
    upstream backend_servers {
        server localhost:8000;
        server localhost:8001;
    }

    server {
        listen 80;

        location /check {
            proxy_pass http://backend_servers/check;
        }

        location /token {
            proxy_pass http://backend_servers/token;
        }
    }
}
```

Python - FastAPI: 460/s



java-test.jmx (/home/uldana/sysdes/oAuth/java-test.jmx) - Apache JMeter (5.6.3)

Summary Report

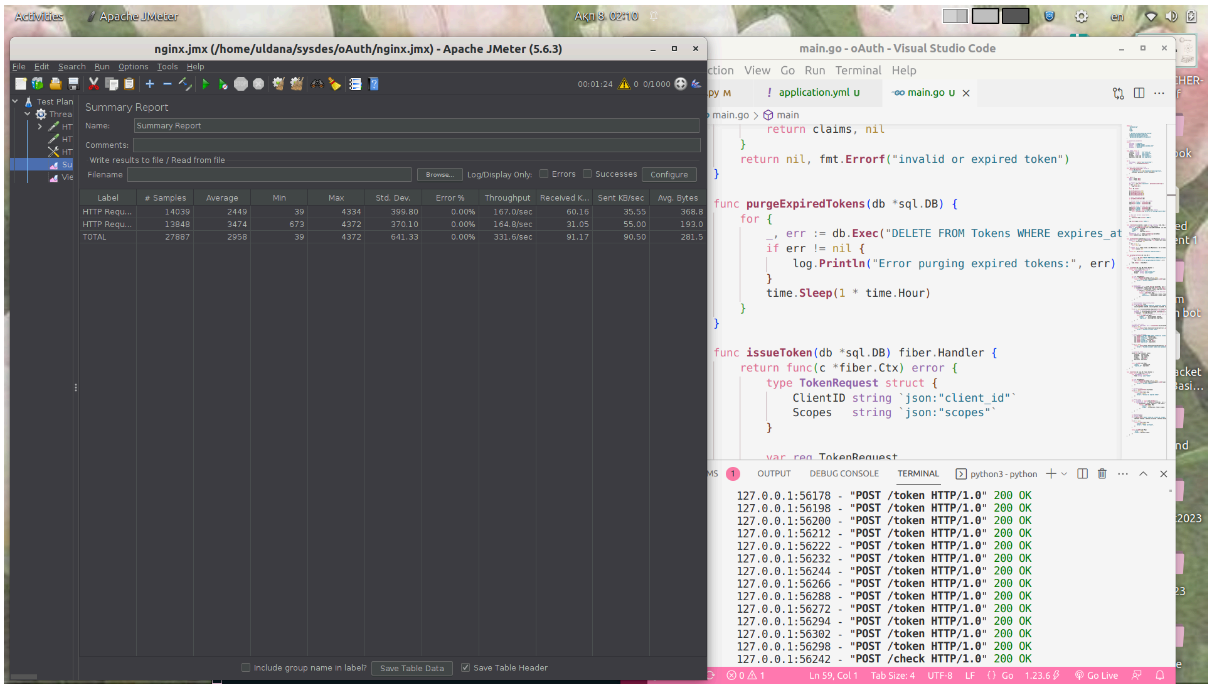
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/...
/token	80923	1955	1051	7748	729.04	0.00%	231.2/sec	74.66
/check	80011	2382	1187	6280	678.55	0.00%	230.2/sec	36.43
TOTAL	160934	2167	1051	7748	736.08	0.00%	459.7/sec	110.84

main.py - oAuth - Visual Studio Code

```python
    # check cache first
    for client_id, cached_token in token_cache.item
        if cached_token["token"] == token:
            return {"valid": True, "scopes": cached

    # If not in cache, check the database
    db = SessionLocal()
    try:
        db_token = db.query(Token).filter(Token.tok
        if not db_token:
            return {"valid": False, "scopes": "Toke
        return {"valid": True, "scopes": db_token.s
    finally:
        db.close()

except JWTError:
    return {"valid": False, "scopes": "Invalid or e
```

```
INFO:     127.0.0.1:46200 - "POST /check HTTP/1.1" 200 OK
INFO:     127.0.0.1:51916 - "POST /check HTTP/1.1" 200 OK
INFO:     127.0.0.1:46396 - "POST /token HTTP/1.1" 200 OK
INFO:     127.0.0.1:46298 - "POST /token HTTP/1.1" 200 OK
INFO:     127.0.0.1:51936 - "POST /check HTTP/1.1" 200 OK
INFO:     127.0.0.1:46564 - "POST /check HTTP/1.1" 200 OK
INFO:     127.0.0.1:45572 - "POST /token HTTP/1.1" 200 OK
INFO:     127.0.0.1:51880 - "POST /check HTTP/1.1" 200 OK
INFO:     127.0.0.1:46346 - "POST /token HTTP/1.1" 200 OK
INFO:     127.0.0.1:46252 - "POST /token HTTP/1.1" 200 OK
INFO:     127.0.0.1:46736 - "POST /check HTTP/1.1" 200 OK
INFO:     127.0.0.1:46458 - "POST /token HTTP/1.1" 200 OK
INFO:     127.0.0.1:46192 - "POST /check HTTP/1.1" 200 OK
```

2 servers -> nginx - 300/s



nginx.jmx (/home/uldana/sysdes/oAuth/nginx.jmx) - Apache JMeter (5.6.3)

Summary Report

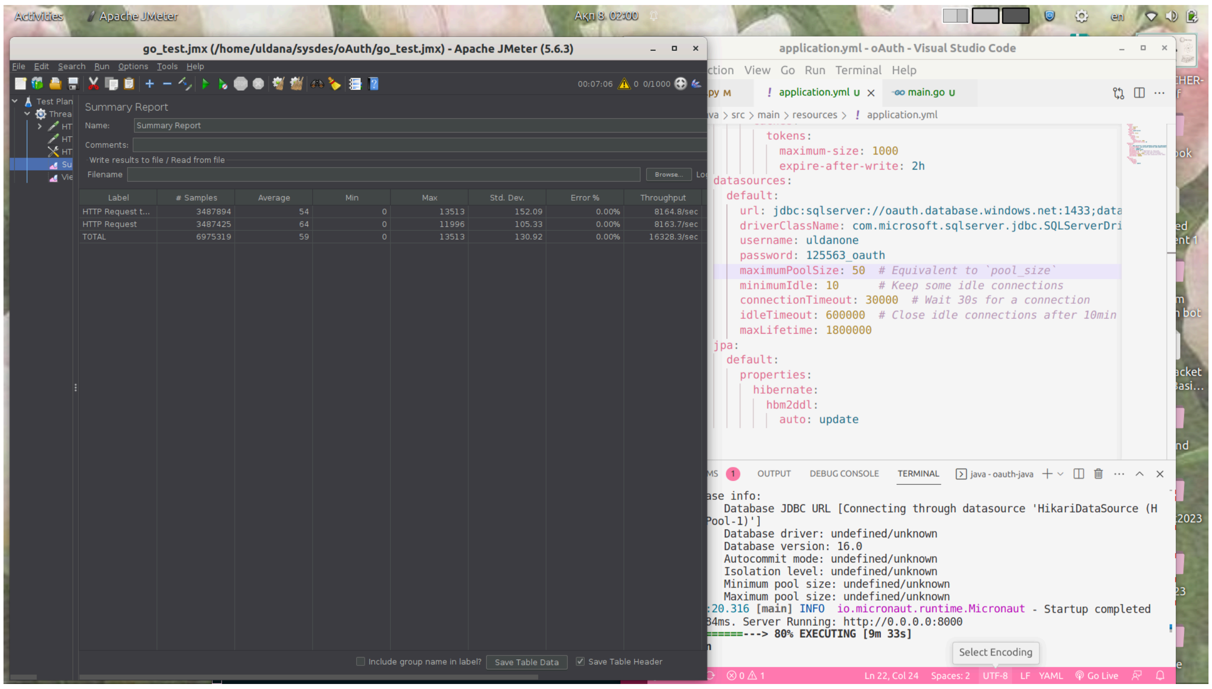
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
HTTP Requ...	14039	2449	39	4334	399.80	0.00%	167.0/sec	60.16	35.55	368.8
HTTP Requ...	13848	3474	673	4372	370.10	0.00%	164.8/sec	31.05	55.00	193.0
TOTAL	27887	2958	39	4372	641.33	0.00%	331.6/sec	91.17	90.50	281.5

main.go - oAuth - Visual Studio Code

```go
        return claims, nil
    }
    return nil, fmt.Errorf("invalid or expired token")
}

func purgeExpiredTokens(db *sql.DB) {
    for {
        _, err := db.Exec("DELETE FROM Tokens WHERE expires_at
        if err != nil {
            log.Println("Error purging expired tokens:", err)
        }
        time.Sleep(1 * time.Hour)
    }
}

func issueToken(db *sql.DB) fiber.Handler {
    return func(c *fiber.Ctx) error {
        type TokenRequest struct {
            ClientID string `json:"client_id"`
            Scopes   string `json:"scopes"`
        }

        var req TokenRequest
```

```
127.0.0.1:56178 - "POST /token HTTP/1.0" 200 OK
127.0.0.1:56198 - "POST /token HTTP/1.0" 200 OK
127.0.0.1:56200 - "POST /token HTTP/1.0" 200 OK
127.0.0.1:56212 - "POST /token HTTP/1.0" 200 OK
127.0.0.1:56222 - "POST /token HTTP/1.0" 200 OK
127.0.0.1:56232 - "POST /token HTTP/1.0" 200 OK
127.0.0.1:56244 - "POST /token HTTP/1.0" 200 OK
127.0.0.1:56266 - "POST /token HTTP/1.0" 200 OK
127.0.0.1:56288 - "POST /token HTTP/1.0" 200 OK
127.0.0.1:56272 - "POST /token HTTP/1.0" 200 OK
127.0.0.1:56294 - "POST /token HTTP/1.0" 200 OK
127.0.0.1:56302 - "POST /token HTTP/1.0" 200 OK
127.0.0.1:56298 - "POST /token HTTP/1.0" 200 OK
127.0.0.1:56242 - "POST /check HTTP/1.0" 200 OK
```

Java - Micronaut: 16300/s - reached that number only eventually…
connection pool -> 50, num of users -> 1000


Activities
Apache JMeter
go_test.jmx (/home/uldana/sysdes/oAuth/go_test.jmx) - Apache JMeter (5.6.3)
File Edit Search Run Options Tools Help
00:07:06 0 0/1000
Test Plan
Summary Report
application.yml - oAuth - Visual Studio Code
application.yml U
main.go U
src > main > resources > application.yml
OUTPUT
DEBUG CONSOLE
TERMINAL
java - oauth-java
Select Encoding
Ln 22, Col 24 Spaces: 2 UTF-8 LF YAML Go Live

Summary Report

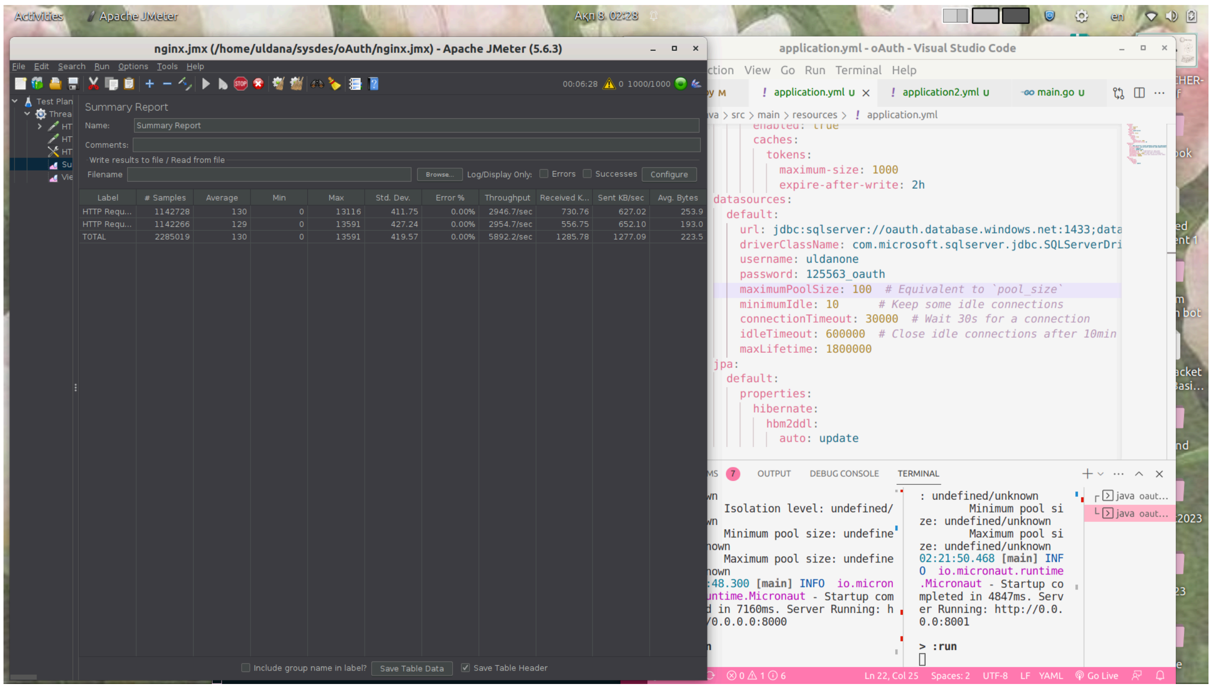
Name: Summary Report

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput
HTTP Request t...	3487894	54	0	13513	152.09	0.00%	8164.8/sec
HTTP Request	3487425	64	0	11996	105.33	0.00%	8163.7/sec
TOTAL	6975319	59	0	13513	130.92	0.00%	16328.3/sec

```yaml
      tokens:
        maximum-size: 1000
        expire-after-write: 2h
datasources:
  default:
    url: jdbc:sqlserver://oauth.database.windows.net:1433;data
    driverClassName: com.microsoft.sqlserver.jdbc.SQLServerDri
    username: uldanone
    password: 125563_oauth
    maximumPoolSize: 50  # Equivalent to `pool_size`
    minimumIdle: 10       # Keep some idle connections
    connectionTimeout: 30000  # Wait 30s for a connection
    idleTimeout: 600000  # Close idle connections after 10min
    maxLifetime: 1800000
jpa:
  default:
    properties:
      hibernate:
        hbm2ddl:
          auto: update
```

```
ase info:
  Database JDBC URL [Connecting through datasource 'HikariDataSource (H
Pool-1)']
  Database driver: undefined/unknown
  Database version: 16.0
  Autocommit mode: undefined/unknown
  Isolation level: undefined/unknown
  Minimum pool size: undefined/unknown
  Maximum pool size: undefined/unknown
:20.316 [main] INFO  io.micronaut.runtime.Micronaut - Startup completed
34ms. Server Running: http://0.0.0.0:8000
======---> 80% EXECUTING [9m 33s]
n
```

2 servers + nginx -> 5800/s



nginx.jmx (/home/uldana/sysdes/oAuth/nginx.jmx) - Apache JMeter (5.6.3)

Summary Report

Name: Summary Report

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
HTTP Requ...	1142728	130	0	13116	411.75	0.00%	2946.7/sec	730.76	627.02	253.9
HTTP Requ...	1142266	129	0	13591	427.24	0.00%	2954.7/sec	556.75	652.10	193.0
TOTAL	2285019	130	0	13591	419.57	0.00%	5892.2/sec	1285.78	1277.09	223.5

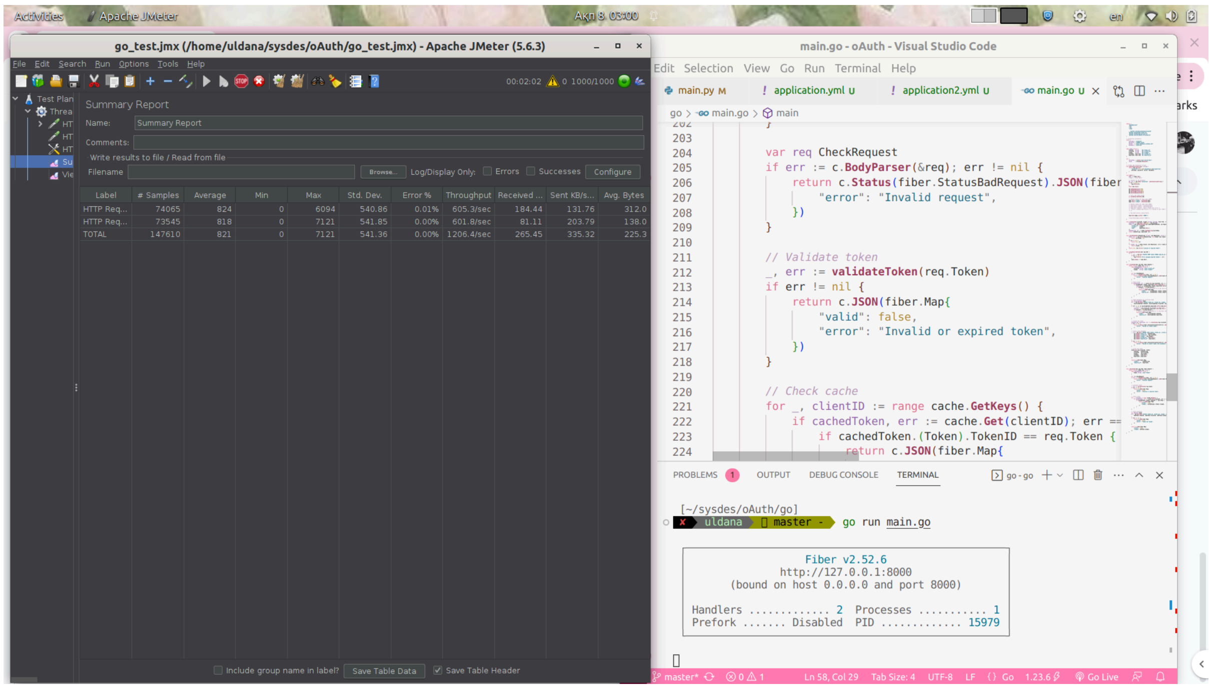
application.yml - oAuth - Visual Studio Code

```yaml
      enabled: true
      caches:
        tokens:
          maximum-size: 1000
          expire-after-write: 2h
datasources:
  default:
    url: jdbc:sqlserver://oauth.database.windows.net:1433;data
    driverClassName: com.microsoft.sqlserver.jdbc.SQLServerDri
    username: uldanone
    password: 125563_oauth
    maximumPoolSize: 100  # Equivalent to `pool_size`
    minimumIdle: 10       # Keep some idle connections
    connectionTimeout: 30000  # Wait 30s for a connection
    idleTimeout: 600000  # Close idle connections after 10min
    maxLifetime: 1800000
jpa:
  default:
    properties:
      hibernate:
        hbm2ddl:
          auto: update
```

```
wn
   Isolation level: undefined/
wn
   Minimum pool size: undefine
nown
   Maximum pool size: undefine
nown
:48.300 [main] INFO  io.micron
untime.Micronaut - Startup com
d in 7160ms. Server Running: h
/0.0.0.0:8000

n
```

```
: undefined/unknown
        Minimum pool si
ze: undefined/unknown
        Maximum pool si
ze: undefined/unknown
02:21:50.468 [main] INF
O  io.micronaut.runtime
.Micronaut - Startup co
mpleted in 4847ms. Serv
er Running: http://0.0.
0.0:8001

> :run
```

Go-fiber -> 1200/s



go_test.jmx (/home/uldana/sysdes/oAuth/go_test.jmx) - Apache JMeter (5.6.3)

Summary Report

Name: Summary Report

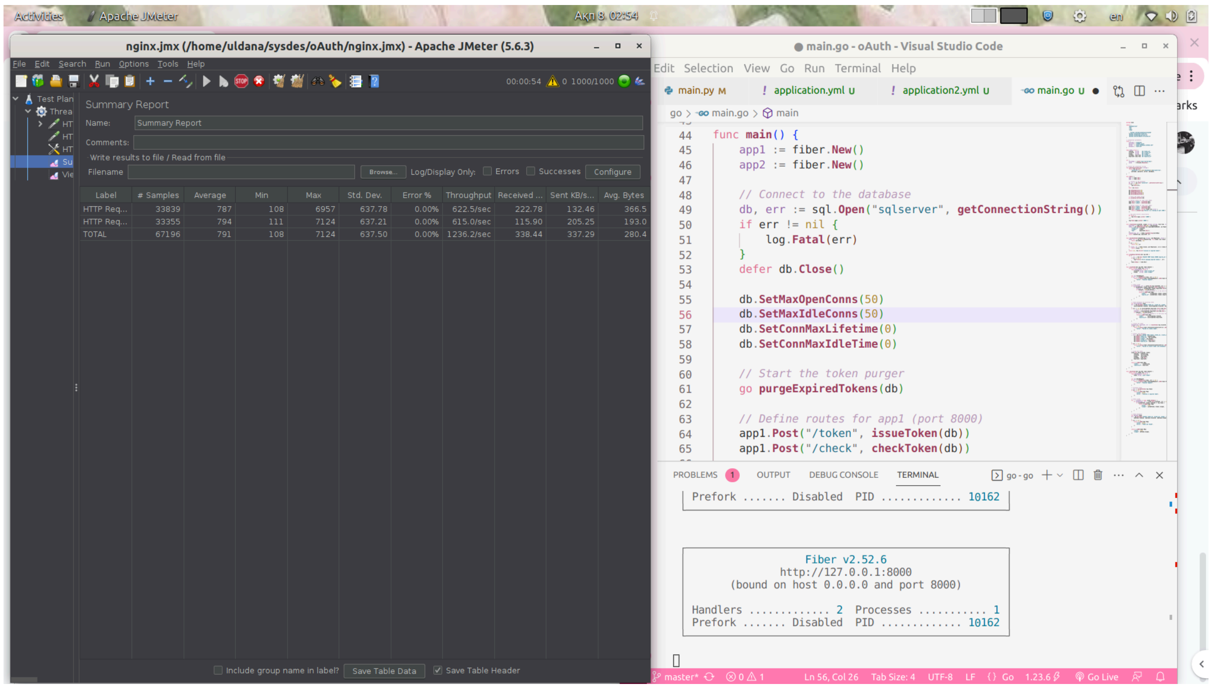
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received ...	Sent KB/s...	Avg. Bytes
HTTP Req...	74065	824	0	6094	540.86	0.01%	605.3/sec	184.44	131.76	312.0
HTTP Req...	73545	818	0	7121	541.85	0.00%	601.8/sec	81.11	203.79	138.0
TOTAL	147610	821	0	7121	541.36	0.00%	1206.4/sec	265.45	335.32	225.3

main.go - oAuth - Visual Studio Code

```go
        }

        var req CheckRequest
        if err := c.BodyParser(&req); err != nil {
            return c.Status(fiber.StatusBadRequest).JSON(fiber
                "error": "Invalid request",
            })
        }

        // Validate token
        _, err := validateToken(req.Token)
        if err != nil {
            return c.JSON(fiber.Map{
                "valid": false,
                "error": "Invalid or expired token",
            })
        }

        // Check cache
        for _, clientID := range cache.GetKeys() {
            if cachedToken, err := cache.Get(clientID); err ==
                if cachedToken.(Token).TokenID == req.Token {
                    return c.JSON(fiber.Map{
```

```
[~/sysdes/oAuth/go]
 ✗  uldana   master -  go run main.go

 ┌───────────────────────────────────────────────────┐
 │                   Fiber v2.52.6                   │
 │               http://127.0.0.1:8000               │
 │       (bound on host 0.0.0.0 and port 8000)       │
 │                                                   │
 │ Handlers ............. 2  Processes ........... 1 │
 │ Prefork ....... Disabled  PID ............. 15979 │
 └───────────────────────────────────────────────────┘
```

2servers + nginx -> 1200/s



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received ...	Sent KB/s...	Avg. Bytes
HTTP Req...	33839	787	108	6957	637.78	0.00%	622.5/sec	222.78	132.46	366.5
HTTP Req...	33355	794	111	7124	637.21	0.00%	615.0/sec	115.90	205.25	193.0
TOTAL	67196	791	108	7124	637.50	0.00%	1236.2/sec	338.44	337.29	280.4

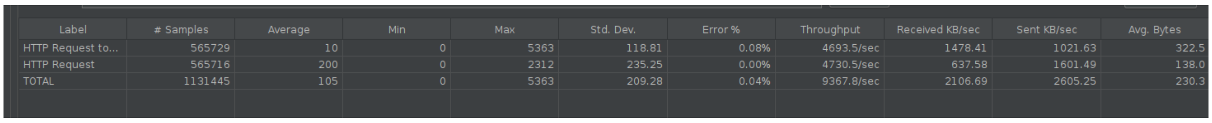
```go
func main() {
	app1 := fiber.New()
	app2 := fiber.New()

	// Connect to the database
	db, err := sql.Open("sqlserver", getConnectionString())
	if err != nil {
		log.Fatal(err)
	}
	defer db.Close()

	db.SetMaxOpenConns(50)
	db.SetMaxIdleConns(50)
	db.SetConnMaxLifetime(0)
	db.SetConnMaxIdleTime(0)

	// Start the token purger
	go purgeExpiredTokens(db)

	// Define routes for app1 (port 8000)
	app1.Post("/token", issueToken(db))
	app1.Post("/check", checkToken(db))
```

```
 Prefork ....... Disabled  PID ............. 10162

                  Fiber v2.52.6
               http://127.0.0.1:8000
       (bound on host 0.0.0.0 and port 8000)

 Handlers ............ 2  Processes ........... 1
 Prefork ....... Disabled  PID ............. 10162
```

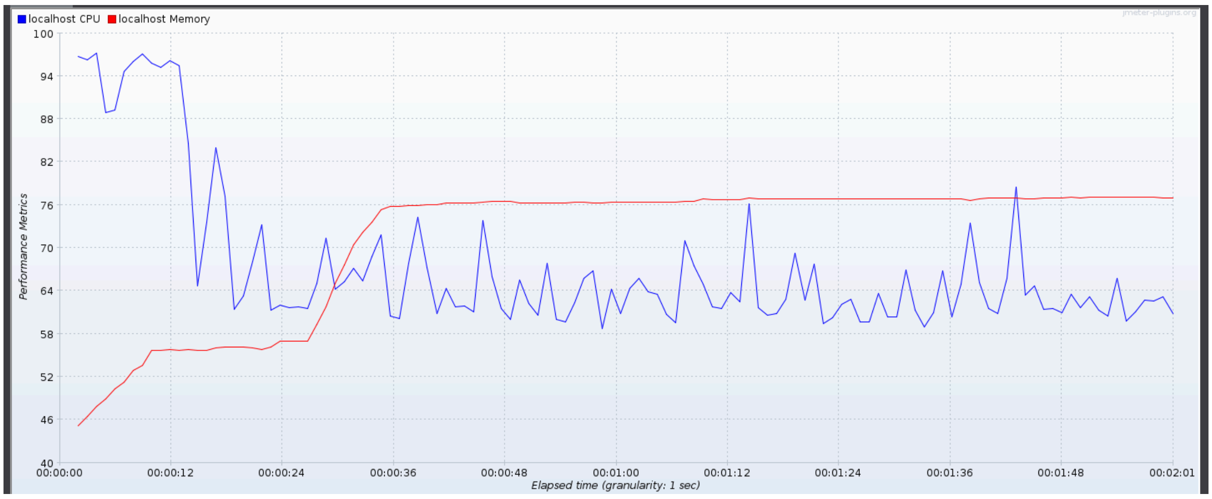
More optimization to be done: setup servers, load balancer, jmeter and connections pool on different machines; replace nginx with other server (1k->10k concurrent connections); improve purger function as low throughput occurred on database with 1000+ rows
Work done: bare minimum implementation of the design discussed on lecture with simple caching + purging system on 3 frameworks, basic setup of nginx as round robin load balancer
Work left: compare in terms of memory/cpu usage

2 minutes, 100 db connections, 1000 users:

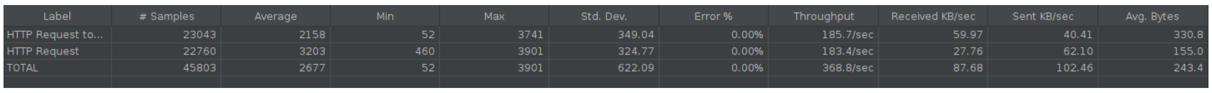
Go



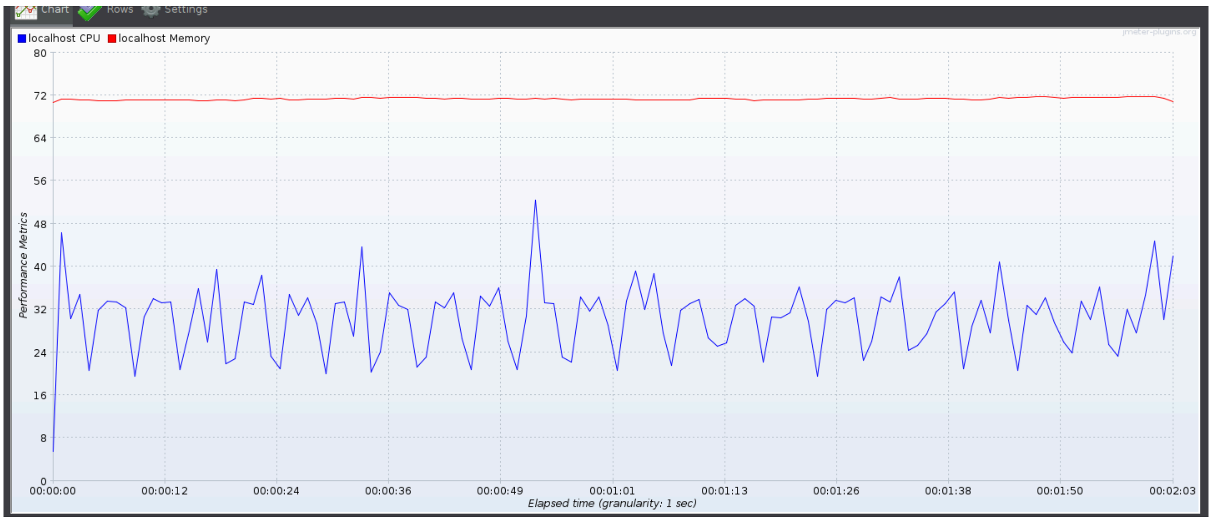
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request to...	565729	10	0	5363	118.81	0.08%	4693.5/sec	1478.41	1021.63	322.5
HTTP Request	565716	200	0	2312	235.25	0.00%	4730.5/sec	637.58	1601.49	138.0
TOTAL	1131445	105	0	5363	209.28	0.04%	9367.8/sec	2106.69	2605.25	230.3



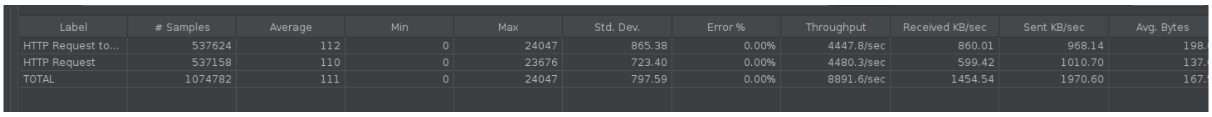
Python



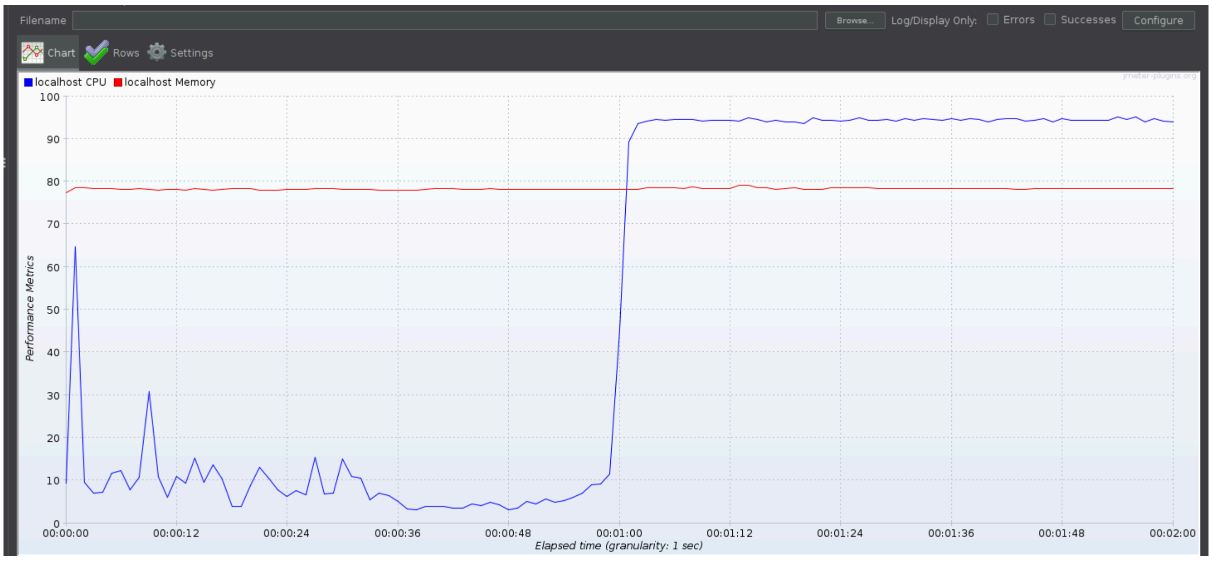
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request to...	23043	2158	52	3741	349.04	0.00%	185.7/sec	59.97	40.41	330.8
HTTP Request	22760	3203	460	3901	324.77	0.00%	183.4/sec	27.76	62.10	155.0
TOTAL	45803	2677	52	3901	622.09	0.00%	368.8/sec	87.68	102.46	243.4



Java



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request to...	537624	112	0	24047	865.38	0.00%	4447.8/sec	860.01	968.14	198.
HTTP Request	537158	110	0	23676	723.40	0.00%	4480.3/sec	599.42	1010.70	137.
TOTAL	1074782	111	0	24047	797.59	0.00%	8891.6/sec	1454.54	1970.60	167.


Filename
Browse...
Log/Display Only:
Errors
Successes
Configure
Chart
Rows
Settings
localhost CPU
localhost Memory
jmeter-plugins.org
Performance Metrics
00:00:00
00:00:12
00:00:24
00:00:36
00:00:48
00:01:00
00:01:12
00:01:24
00:01:36
00:01:48
00:02:00
Elapsed time (granularity: 1 sec)