

Selected database products overview

Installing Docker

- OS specific instructions at [Install Docker Engine](#).
- Windows: Make sure to install Windows Subsystem for Linux version 2 (WSL 2). Install [Ubuntu](#) from Microsoft's App store and follow [Docker Desktop WSL 2 backend](#).

Selected database products overview
⇒ OpenIdap

Why LDAP?

- **Standardized** protocol like [HTTP](#), [SMB](#) and others.
- “Glue” connecting an organization's applications and services.
 - Backend data for authentication / SSO, PKI, network FS etc.
 - Meta data repository for users, organisational units, network nodes.

LDAP introduction

- Read the introductory slides at [the section called “LDAP”](#) as well.

Selected database products overview

⇒ Openldap

⇒ Installation

Running a Docker container

```
docker run --detach ❶ \  
  --name openldap ❷ \  
  -p 389:389 \ ❸ \  
  --env LDAP_ORGANISATION="Betrayers heaven" \ ❹ \  
  --env LDAP_TLS=false ❺ \  
  --env LDAP_DOMAIN="betrayer.com" ❻ \  
  --env LDAP_ADMIN_PASSWORD="secret" ❼ \  
  --env LDAP_CONFIG_PASSWORD="secret" ❽ \  
  --volume ~/OpenLdap/Data:/var/lib/ldap ❾ \  
  --volume ~/OpenLdap/Config:/etc/ldap/slapd.d ❾ \  
osixia/openldap:1.4.0 ❿
```

Using docker-compose

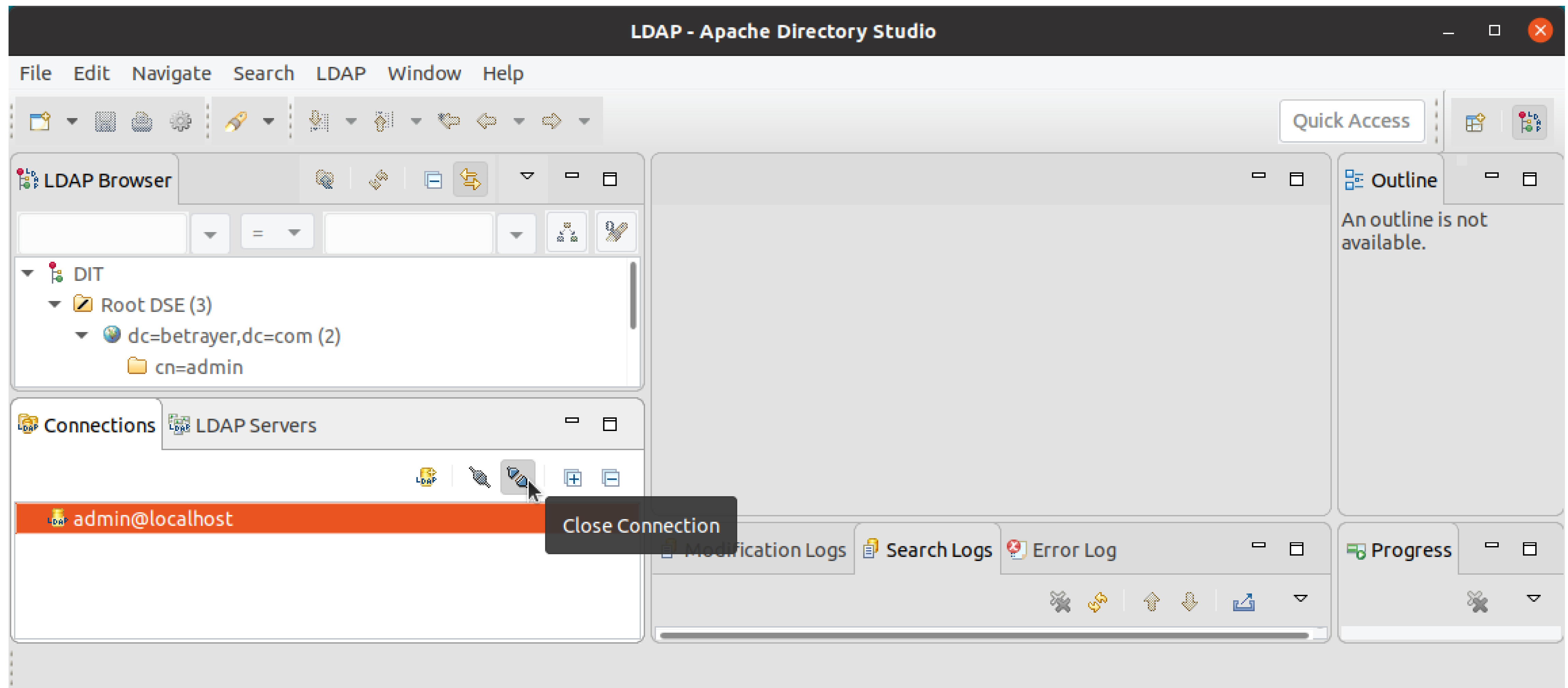
```
version: '3.7'

services:
  openldap:
    image: osixia/openldap:1.4.0
    container_name: openldap
    restart: always
    environment:
      LDAP_ORGANISATION: "Betrayers heaven"
      LDAP_TLS: "false"
      LDAP_DOMAIN: "betrayer.com"
      LDAP_ADMIN_PASSWORD: "secret"
      LDAP_CONFIG_PASSWORD: "secret"
    ports:
      - 389:389
    volumes:
```

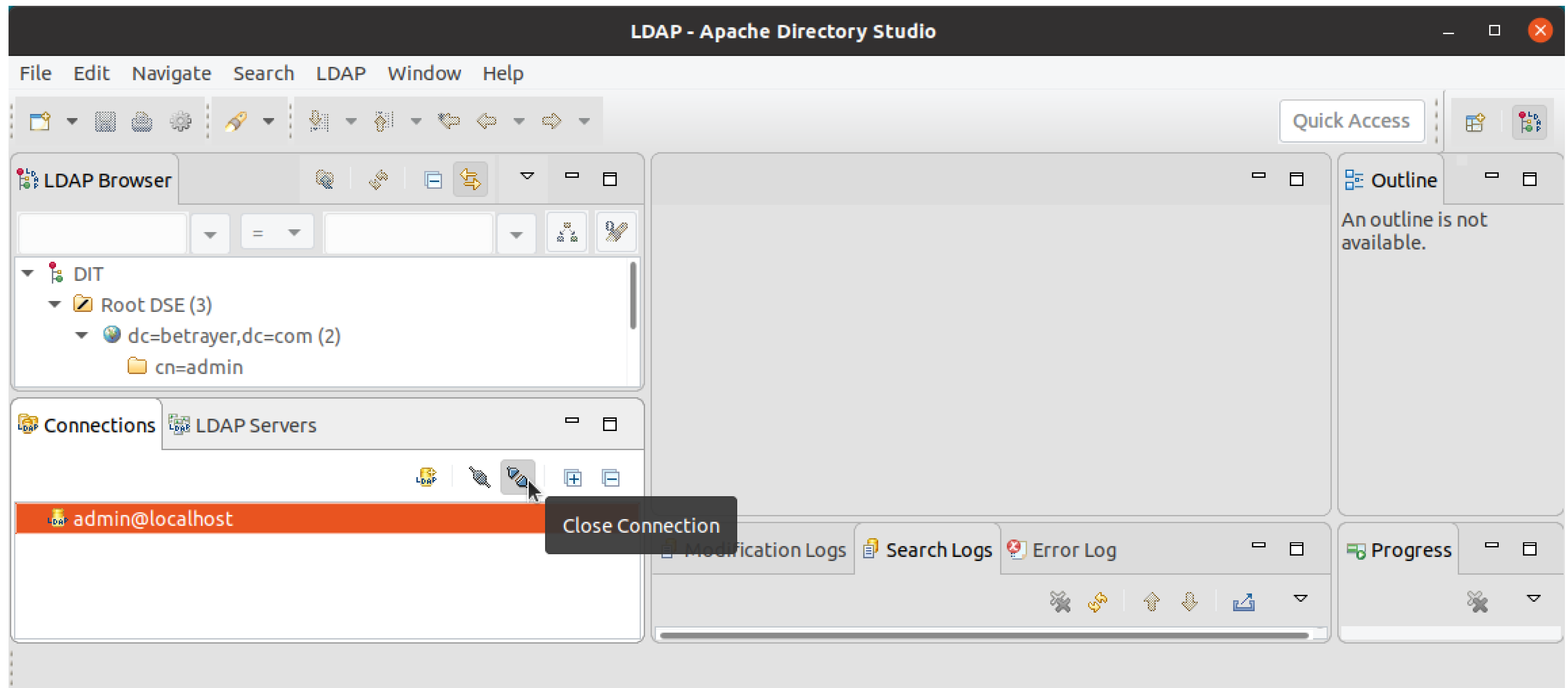
Installing Apache Directory Studio

- Download and install [Apache Directory Studio](#).
- Configure access to your local docker container.

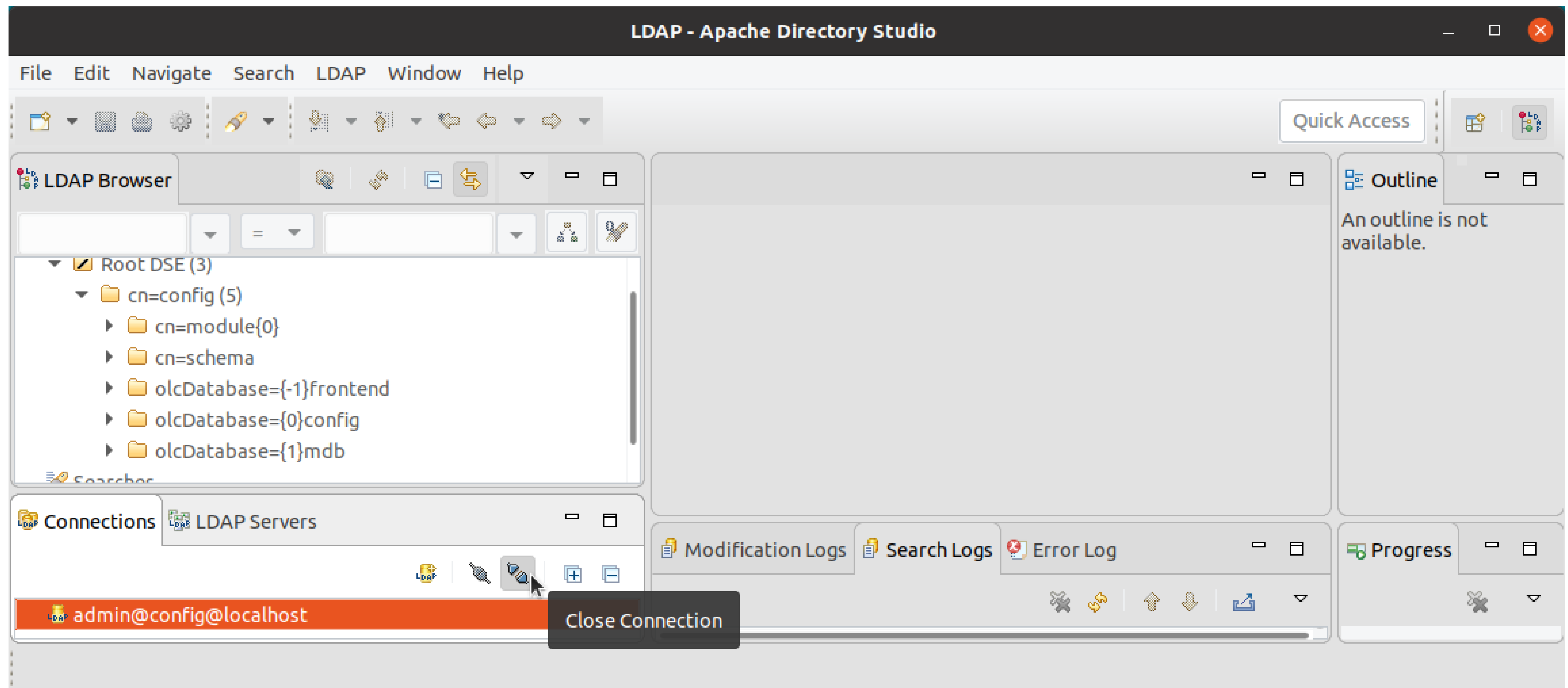
Administrator access to your DIT



Administrator access to your server's data tree



Administrator access to your server's configuration



Selected database products overview

⇒ Openldap

⇒ Features

Terminology

- **DIT**: Document information tree
- **DN**: An entries distinguished name. Unique identifier.
- **RDN**: Relative distinguished name. Unique identifier with respect to a given context node.
- Bind operation: Connect to an **LDAP** service.
 1. anonymous
 2. authenticated

Selected database products overview

⇒ OpenIdap

⇒ Features

⇒ CRUD

Adding an entry

```
dn: uid=smith ①,dc=betrayer,dc=com ②
changetype: add ③
objectClass: inetOrgPerson ④
objectClass: organizationalPerson ④
objectClass: Person ④
objectClass: top ④
uid: smith ⑤
cn: Jill Smith ⑥
sn: Smith ⑥
```

Adding a new attribute

Operation	Result
<pre>dn: uid=smith,dc=betrayer,dc=com ① changetype: modify ② add: description ③ description: New employee ④</pre>	<pre>dn: uid=smith,dc=betrayer,dc=com objectClass: inetOrgPerson objectClass: organizationalPerson objectClass: person objectClass: top cn: Jill Smith sn: Smith uid: smith description: New employee</pre>

Replacing an attribute value

Operation	Result
dn: uid=smith,dc=betrayer,dc=com 1 changetype: modify 2 replace: description 3 description: Long term employee 4	dn: uid=smith,dc=betrayer,dc=com objectClass: inetOrgPerson objectClass: organizationalPerson objectClass: person objectClass: top cn: Jill Smith sn: Smith uid: smith description: Long term employee

Deleting an attribute entirely

Base state	Operation	Result
<pre>dn: uid=smith,dc=betrayer,dc=com ... uid: smith commonName: Jill Smith surname: Smith description: Long term employee</pre>	<pre>dn: uid=smith,dc=... changetype: modify delete: description</pre>	<pre>dn: uid=smith,dc=betrayer,dc=com ... uid: smith commonName: Jill Smith surname: Smith</pre>

Multi valued attributes

Operation	Result
<pre>dn: uid=smith,dc=betrayer,dc=com changetype: modify add: mail ① mail: smith@company.com ② mail: jsmith@privateaccount.org ③</pre>	<pre>dn: uid=smith,dc=betrayer,dc=com ... sn: Smith mail: jsmith@privateaccount.org ② mail: smith@company.com ③</pre>

Set semantics of multivalued attributes

Base state	Operation
dn: uid=smith,dc=betrayer,dc=com objectClass: inetOrgPerson objectClass: organizationalPerson objectClass: person objectClass: top uid: smith commonName: Jill Smith surname: Smith	dn: uid=smith,dc=betrayer,dc=com changetype: modify add: mail ① mail: smith@company.com ② mail: jsmith@privateaccount.org ③ mail: smith@company.com ④ ERR_13207_VALUE_ALREADY_EXISTS ④ The value 'smith@company.com' already exists in the attribute (mail)

Deleting selected attribute values

Base state	Operation	Result
dn: uid=smith,dc=betrayer,dc=com ... cn: Jill Smith mail: jsmith@privateaccount.org mail: smith@company.com mail: anonymous@keeput.org	dn: uid=smith,dc=... changetype: modify delete: mail mail: smith@company.com mail: anonymous@keeput.org	dn: uid=smith,dc=betrayer,dc=com ... cn: Jill Smith mail: jsmith@privateaccount.org

Selected database products overview

⇒ OpenIdap

⇒ Features

⇒ Query

Query scope

- BaseObject
- SingleLevel
- WholeSubtree
- SubordinateSubtree (not yet officially standardized)

Query filter

- Presence: (cn=*)
- Equality: (uid=xy123)
- Comparison: (1000 < uidNumber)
- Substring: (surname=K*)
- Approximate Match: (sn~**=**gok)
(matches Gack, Keck, Kubiac,
Goik, Kabbeck, Nguyen Quoc, Gubic,
Koc)
- AND: (**&**(5 < uidNumber)(sn=*k*)
(gidNumber=1000))
- OR: (**|**(5 < uidNumber)(sn=*k*)
(gidNumber=1000))
- NOT: (**!**(5 < uidNumber))
- [Extensible match filters](#), see [Extensible Match Search Filter](#) as well.

Selected database products overview

⇒ Openldap

⇒ Features

⇒ Schema

Schema support

- Objectclass definitions
- Objectclass inheritance.
Types:
 - Abstract
 - Structural
 - Auxiliary
- Dynamic instance schema changes
- Single and multivalued attributes
- Attribute data types
- Matching rules (Case sensitive / insensitive, string match, numeric)

Selected database products overview

⇒ Openldap

⇒ Features

⇒ Data access control

Implementations

```
access to attrs=matrikelNr
    by dn="uid=goik,ou=userlist,dc=hdm-stuttgart,dc=de" read
    by dn="uid=kuhn,ou=userlist,dc=hdm-stuttgart,dc=de" read
    by self read
    by * none

access to attrs=userPassword,shadowLastChange,passwordClear
    by dn="uid=Administrator,ou=people,ou=MI,ou=domainlist,dc=hdm-stuttgart,dc=de" read
    by anonymous auth
    by * none
```

Selected database products overview

- ⇒ Openldap

- ⇒ Features

- ⇒ API support

Implementations

- JNDI
- Ldapive

Selected database products overview

⇒ Openldap

⇒ Exercises

1. Work through the exercises the section called “Browse an existing LDAP Server” and the section called “Populating your DIT.” to the section called “Extending an existing entry”.

Tip

When logging in as a non - admin user i.e. using a bind DN like `uid=petra,ou=MIB,ou=MI,dc=betray,dc=com` you will not be able to browse your tree. This action requires a permission setting to be changed in `olcDatabase={1}mdb,cn=config` of your server's configuration tree. Follow these steps:

1. Log in to your server's configuration using `cn=admin,cn=config` as in Figure 797, “Administrator access to your server's configuration”.
2. Select your database backend node below `cn=config`.
3. Replace:

```
to * by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth manage by * break
```

By:

```
to * by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth manage by * read
```

Selected database products overview
⇒ MongoDB

Why MongoDB?

- Document oriented (hierarchy support).
- Horizontal scaling (Sharding)
- Large user community
- Major programming languages API support.
- Open Source

Selected database products overview

⇒ MongoDB

⇒ Installation

Running a Docker container

```
docker run -d \  
  --name localMongoDb ❶ \  
  -e MONGO_INITDB_ROOT_USERNAME=admin ❷ \  
  -e MONGO_INITDB_ROOT_PASSWORD=secret ❷ \  
  -e MONGO_INITDB_DATABASE=admin ❸ \  
  -v ~/Data/Mongo:/data/db ❹ \  
  -p 27017:27017 ❺ \  
  mongo:4.4.1 ❻
```

Using docker-compose

docker-compose.yml

```
version: '3.7'

services:
  mongodb:
    image: mongo:4.4.1
    container_name: mongodb
    restart: always
    environment:
      MONGO_INITDB_ROOT_USERNAME: admin
      MONGO_INITDB_ROOT_PASSWORD: secret
    ports:
      - 27017:27017
    volumes:
      - ./mongo-init.js:/docker-entrypoint-initdb.d/mongo-init.js:ro
```

mongo-init.js

```
db.createUser(
  {
    user: "explorer",
    pwd: "secret",
    roles: [
      {
        role: "readWrite"
        db: "exploredb"
      }
    ],
    passwordDigestor: "server"
  }
);
```

```
docker-compose up --build -d
```

Manual user creation

```
> mongo -u admin -p secret admin
...
db.createUser(
...   {
...     user: "explorer",
...     pwd: "secret",
...     roles: [
...       {
...         role: "readWrite",
...         db: "exploredb"
...       }
...     ],
...     passwordDigestor: "server"
...   }
... );
Successfully added user: { "user" : "explorer"...
```

Log in as user explorer

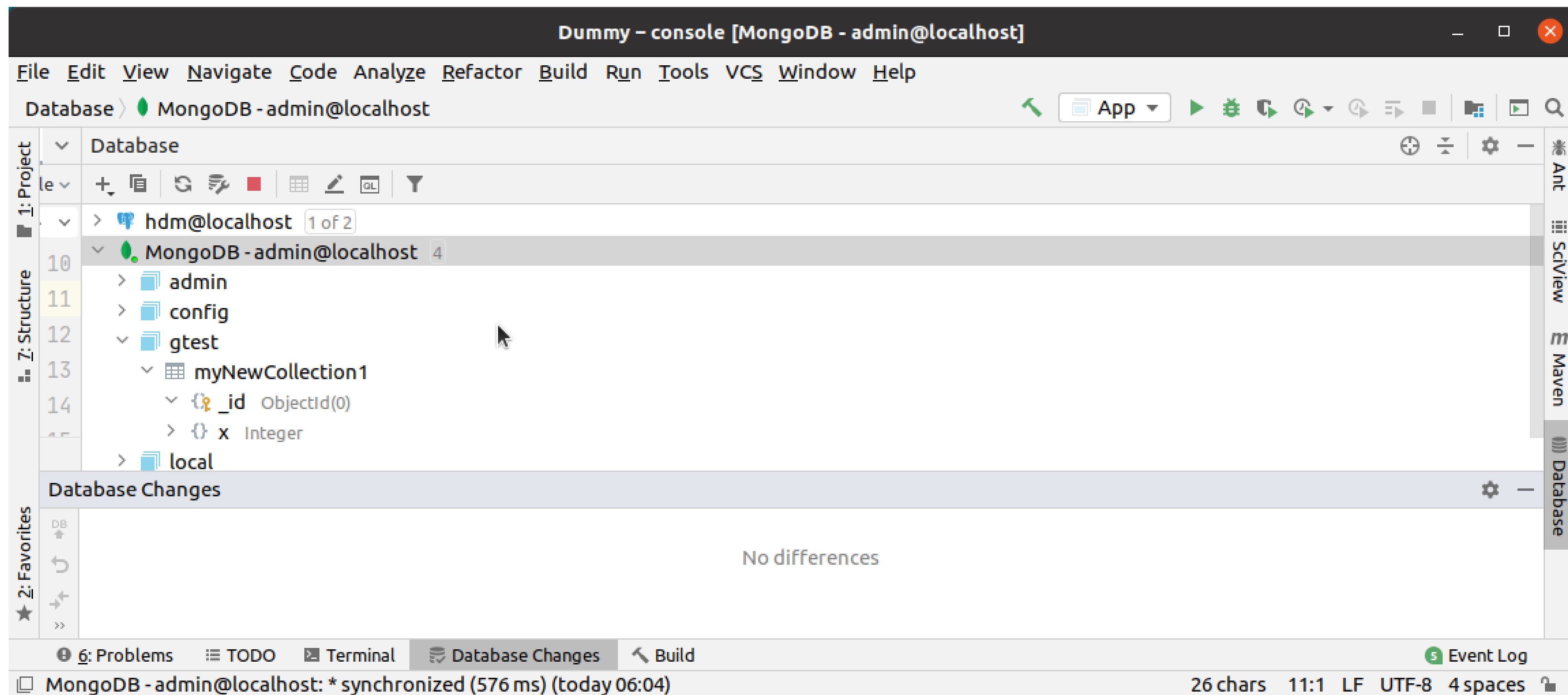
```
> mongo -u explorer -p secret admin
...
> use exploredb
switched to db exploredb

> db.user.insert(
...   { cname: "Eve Gardener",
...     uid: "gardener",
...     email: "gardener@betrayer.com"
...   }
... )
WriteResult({ "nInserted" : 1 })
>
> db.user.find()
{ "_id" : ObjectId("5fa1c79d661a55242658f135"),
  "cname" : "Eve Gardener", "uid" : "gardener", "email" : "gardener@betrayer.com" }
```

Using IntelliJ

- View --> Tool Windows --> Database
- Data Source --> MongoDB

Idea show all databases



Selected database products overview

- ⇒ MongoDB

- ⇒ Features

- MongoDB Server
 - **Database**
 - **Collection** : Similar to tables in **SQL**.
 - **Document**: Similar to records in **SQL**.

```
> mongo -u explorer -p secret admin
> use exploredb
> db.user.insert(
    { cname: "Eve Gardener",
      uid: "gardener",
      email: "gardener@betrayer.com"
    }
  )
```

Selected database products overview

- ⇒ MongoDB

- ⇒ Features

- ⇒ CRUD

Adding a document

Code	Result
<pre>db.group.insert({ cname: "All users", gid: "users", })</pre>	<pre>db.group.find() [{ "_id": {"\$oid": "5fa3035932b87a0c60a6ed1a"}, "cname": "New users", "gid": "users" }]</pre>

Updating attributes

Code	Result
<pre>db.group.update({_id: ObjectId("5fa3035932b87a0c60a6ed1a")}, { \$set: { cname: "New users" ①, gidNumber: 1000 ② } })</pre>	<pre>db.group.find() [{ "_id": {"\$oid": "5fa3035932b87a0c60a6ed1a"}, cname: "New users" ①, "gid": "users", "gidNumber": 1000 ② }]</pre>

Deleting a document

Code	Result
<pre>db.group.deleteOne({"_id": ObjectId("5fa3035932b87a0c60a6ed1a")}));</pre>	<pre>[{ "acknowledged": true, "deletedCount": 1 }]</pre>

Deleting multiple documents

Code	Result
<pre>db.group.deleteMany({});</pre>	<pre>[{ "acknowledged": true, "deletedCount": 23 }]</pre>

Multi valued attributes

Set semantics of multivalued attributes

Deleting selected attribute values

Deleting an attribute

Code	Result
<pre>db.group.update({_id: ObjectId("5fa3035932b87a0c60a6ed1a")}, { \$unset: { gidNumber: 42 ① } })</pre>	<pre>db.group.find() [{ "_id": {"\$oid": "5fa3035932b87a0c60a6ed1a"}, "cname": "My users", "gid": "users" }]</pre>

Selected database products overview

- ⇒ MongoDB

- ⇒ Features

- ⇒ Query

Query filter

ToDo

Selected database products overview

- ⇒ MongoDB

- ⇒ Features

- ⇒ Schema

Schema validation support

```
db.runCommand( {  
  collMod: "group",  
  validator: { $jsonSchema: {  
    bsonType: "object",  
    required: [ "cname", "gid" ],  
    properties: {  
      cname: {  
        bsonType: "string",  
        description:  
          "A group's common name"  
      },  
      gid: {  
        bsonType: "string",  
        description:  
          "A group's short name"  
      },  
    },  
    validationLevel: "moderate"  
  }  
})
```

```
      gid: {  
        bsonType: "string",  
        description:  
          "A group's short name"  
      },  
    },  
    validationLevel: "moderate"  
  }  
})
```

Violating required field

```
db.group.insert(  
  {  
    cname: "Extra users"  
  }  
)
```

```
... Bulk write operation error on server localhost:27017.  
Write errors: [BulkWriteError{index=0, code=121,  
  message='Document failed validation', details={}}].
```

Schema types

See [BSON Types](#) for reference.

Enforcing unique keys

```
db.group.createIndex({
  "cname": 1
},
{
  unique: true
})

db.group.insert({cname: "Extra users"
  ...
})
```

```
com.mongodb.MongoBulkWriteException:
Bulk write operation error on server localhost:27017.
Write errors: [BulkWriteError{index=0, code=11000,
message='E11000 duplicate key error collection:
exploredb.group index: cname_1 dup key:
{ cname: "Extra users" }, details={}}].
```

On the downside

- No way to enforce referential integrity rules.

Selected database products overview

- ⇒ MongoDB

- ⇒ Features

- ⇒ Data access control

Implementations

Selected database products overview

- ⇒ MongoDB

- ⇒ Features

- ⇒ API support

Implementations

-
-

Selected database products overview

- ⇒ MongoDB

- ⇒ High performance sharding cluster

Sharding rationale

- Problem: Large datasets / high throughput
- Two alternatives:
 - Vertical scaling: RAM, cpu,...
 - Horizontal scaling: Load distribution by multiple nodes.

Sharding rationale

See [sharded-cluster](#) for details.

Selected database products overview

- ⇒ MongoDB

- ⇒ Exercises

Exercises

1.

2.