

# 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.

# 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 Chapter 28, *LDAP* as well.

# 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 ❿\  
  osixia/openldap:1.4.0 ⓫
```

# 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:

    - ~/OpenLdap/Data:/var/lib/ldap

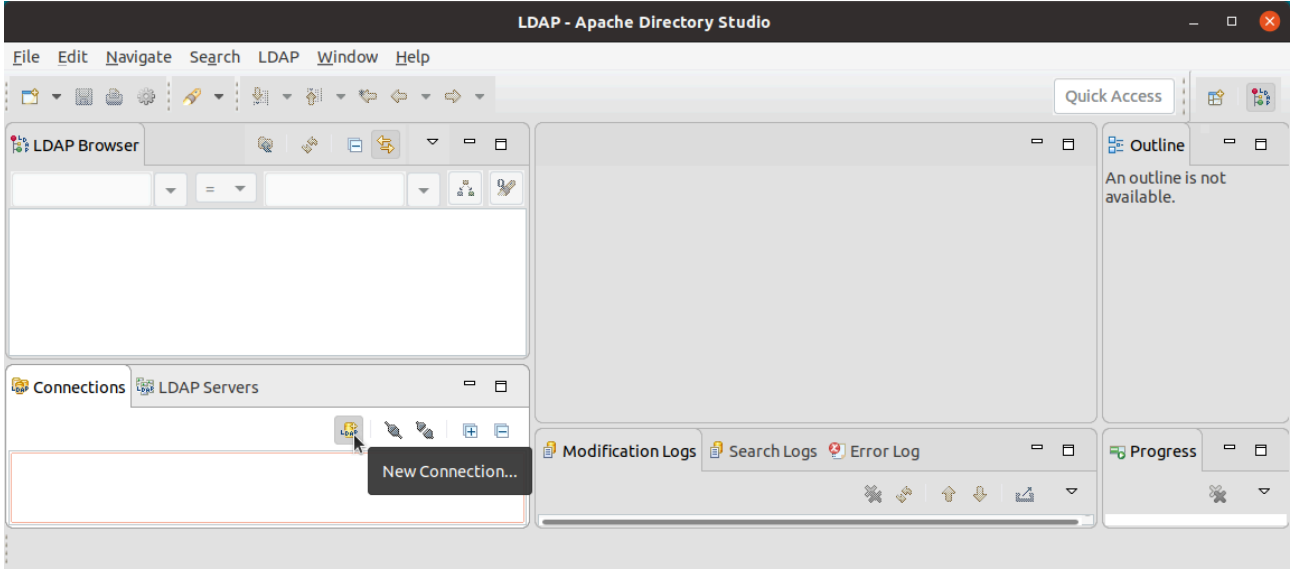
    - ~/OpenLdap/Config:/etc/ldap/slapd.d

# 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 DIT

New LDAP Connection

**Network Parameter**  
Please enter connection name and network parameters.

Connection name: admin@localhost

Network Parameter

Hostname: localhost

Port: 389

Connection timeout (s): 30

Encryption method: No encryption

Server certificates for LDAP connections can be managed in the ['Certificate Validation'](#) preference page.

Provider: Apache Directory LDAP Client API

Check Network Parameter

Read-Only (prevents any add, delete, modify or rename operation)

? < Back Next > Cancel Finish



# Administrator access to your DIT

The screenshot shows the 'New LDAP Connection' wizard. The 'Network Parameter' section is active, with the following fields filled: Connection name: admin@localhost, Hostname: localhost, Port: 389, Connection timeout (s): 30, Encryption method: No encryption, Server certificates for LDAP: (empty), and Provider: Apache Directory LDAP Client API. A 'Check Network Parameter' button is visible in the bottom right of the wizard. A modal dialog box titled 'Check Network Parameter' is overlaid on the wizard, displaying an information icon and the message 'The connection was established successfully.' with an 'OK' button.

**New LDAP Connection**

Please enter connection name and network parameters.

Connection name: admin@localhost

Network Parameter

Hostname: localhost

Port: 389

Connection timeout (s): 30

Encryption method: No encryption

Server certificates for LDAP: (empty)

Provider: Apache Directory LDAP Client API

Check Network Parameter

Read-Only (prevents any add, delete, modify or rename operation)

< Back   Next >   Cancel   Finish

**Check Network Parameter**

The connection was established successfully.

OK

# Administrator access to your DIT

New LDAP Connection

**Network Parameter**  
Please enter connection name and network parameters.

Connection name: admin@localhost

Network Parameter

Hostname: localhost

Port: 389

Connection timeout (s): 30

Encryption method: No encryption

Server certificates for LDAP connections can be managed in the ['Certificate Validation'](#) preference page.

Provider: Apache Directory LDAP Client API

Check Network Parameter

Read-Only (prevents any add, delete, modify or rename operation)

< Back Next > Cancel Finish

# Administrator access to your DIT

New LDAP Connection

**Authentication**  
Please select an authentication method and input authentication data.

Authentication Method  
Simple Authentication

Authentication Parameter

Bind DN or user: cn=admin,dc=betrayer,dc=com

Authorization ID (SASL): SASL PLAIN only

Bind password: \*\*\*\*\*

Save password

Check Authentication

▸ SASL Settings  
▸ Kerberos Settings

? < Back Next > Cancel Finish

# Administrator access to your DIT

The screenshot shows a 'New LDAP Connection' dialog box with the following fields and settings:

- Authentication Method:** Simple Authentication
- Authentication Parameter:** (empty)
- Bind DN or user:** cn=admin,dc=betrayer,dc=com
- Authorization ID (SASL):** SASL PLAIN only
- Bind password:** (masked with asterisks)
- Save password
- SASL Settings** (expanded)
- Kerberos Settings** (collapsed)

A 'Check Authentication' dialog box is overlaid on top, displaying the message: 'The authentication was successful.' with an 'OK' button.

At the bottom of the main dialog, there are navigation buttons: '< Back', 'Next >', 'Cancel', and 'Finish'. A 'Check Authentication' button is also present near the password field.

# Administrator access to your DIT

New LDAP Connection

**Authentication**  
Please select an authentication method and input authentication data.

Authentication Method: Simple Authentication

Authentication Parameter: Bind DN or user: cn=admin,dc=betrayer,dc=com

Authorization ID (SASL): SASL PLAIN only

Bind password: \*\*\*\*\*

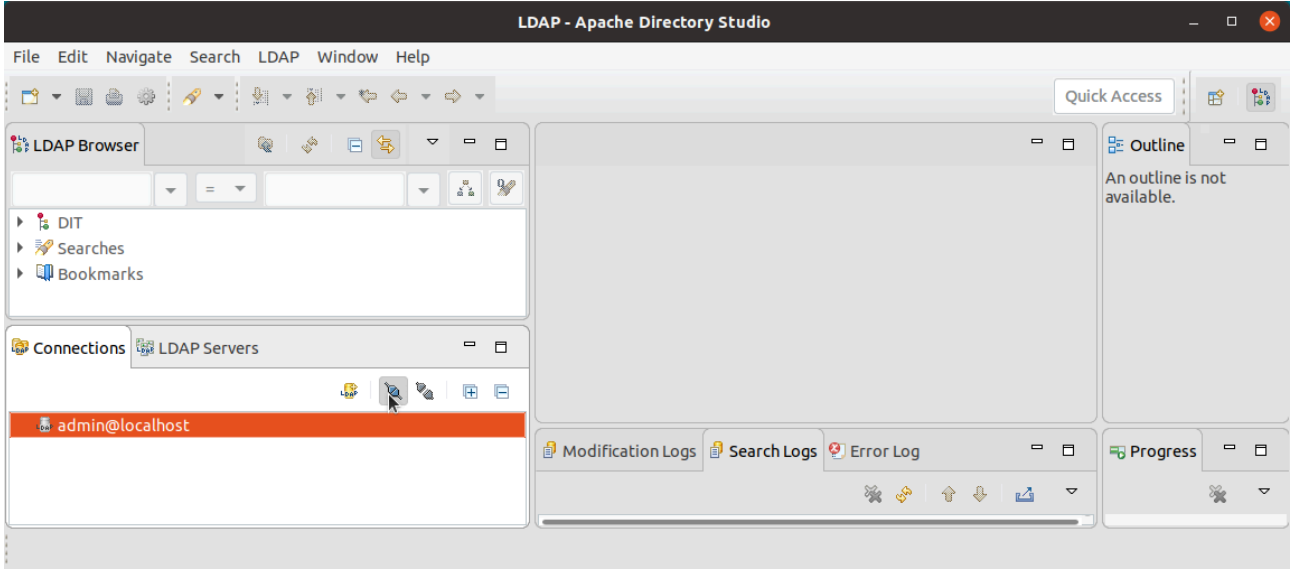
Save password

Check Authentication

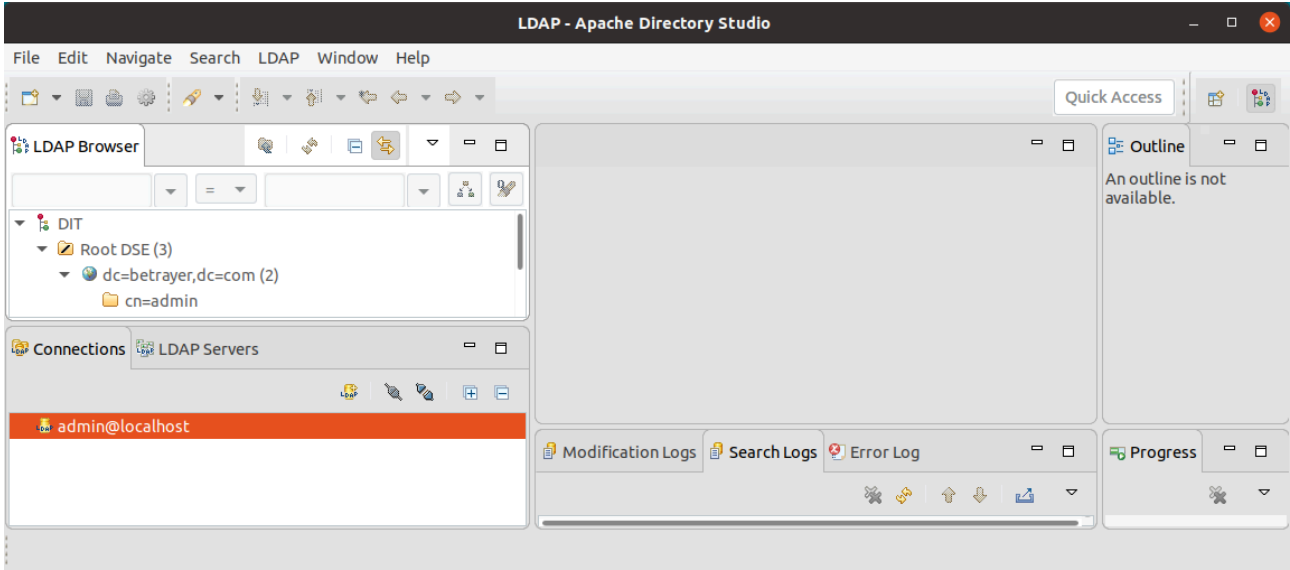
▸ SASL Settings  
▸ Kerberos Settings

< Back   Next >   Cancel   Finish

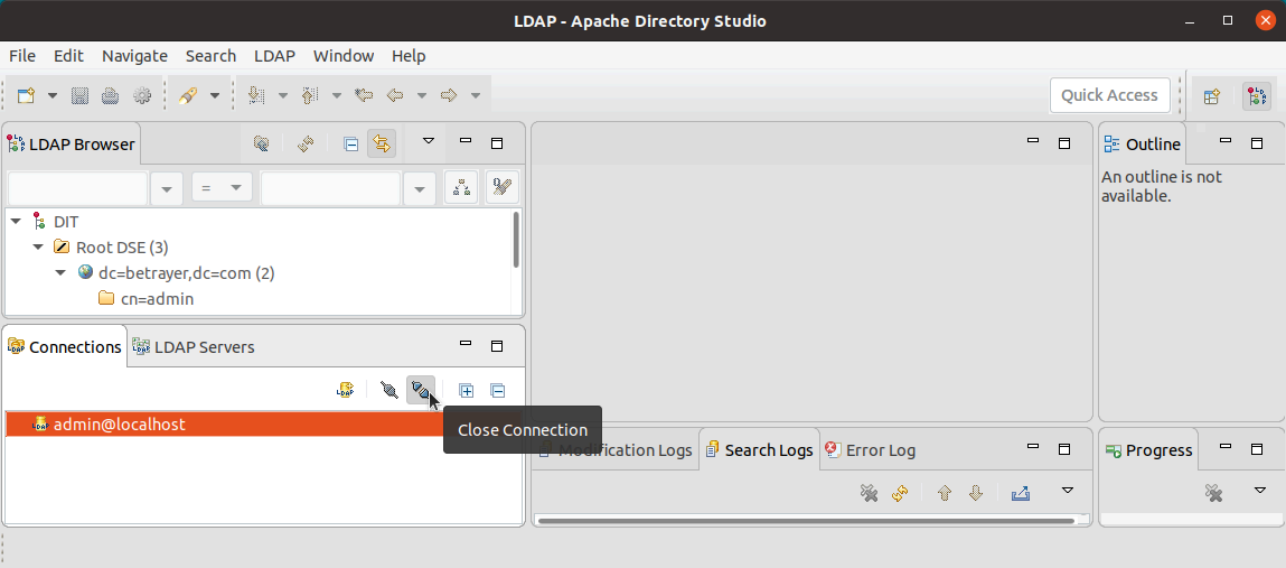
# Administrator access to your DIT



# Administrator access to your DIT

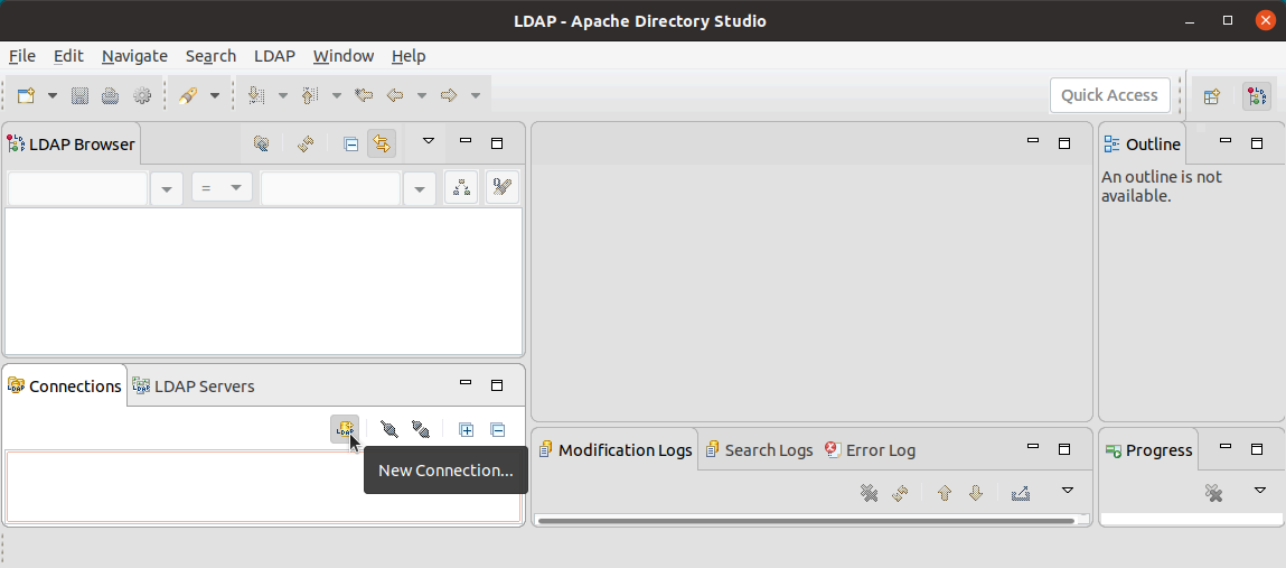


# Administrator access to your DIT





# Administrator access to your server's data tree



# Administrator access to your server's data tree

New LDAP Connection

**Network Parameter**  
Please enter connection name and network parameters.

Connection name: admin@localhost

Network Parameter

Hostname: localhost

Port: 389

Connection timeout (s): 30

Encryption method: No encryption

Server certificates for LDAP connections can be managed in the ['Certificate Validation'](#) preference page.

Provider: Apache Directory LDAP Client API

Check Network Parameter

Read-Only (prevents any add, delete, modify or rename operation)

< Back   Next >   Cancel   Finish

# Administrator access to your server's data tree

The screenshot shows a 'New LDAP Connection' dialog box with the following fields and options:

- Connection name: admin@localhost
- Network Parameter: localhost
- Port: 389
- Connection timeout (s): 30
- Encryption method: No encryption
- Server certificates for LDAP connection: (empty)
- Provider: Apache Directory LDAP Client API
- Read-Only (prevents any add, delete, modify or rename operation):

A 'Check Network Parameter' button is located at the bottom right of the dialog. An information dialog box is overlaid on top, displaying the message: 'The connection was established successfully.' with an 'OK' button.

# Administrator access to your server's data tree

New LDAP Connection

**Network Parameter**  
Please enter connection name and network parameters.

Connection name: admin@localhost

Network Parameter

Hostname: localhost

Port: 389

Connection timeout (s): 30

Encryption method: No encryption

Server certificates for LDAP connections can be managed in the ['Certificate Validation'](#) preference page.

Provider: Apache Directory LDAP Client API

Check Network Parameter

Read-Only (prevents any add, delete, modify or rename operation)

< Back Next > Cancel Finish

# Administrator access to your server's data tree

New LDAP Connection

**Authentication**  
Please select an authentication method and input authentication data.

Authentication Method: Simple Authentication

Authentication Parameter

Bind DN or user: cn=admin,dc=betrayer,dc=com

Authorization ID (SASL): SASL PLAIN only

Bind password: \*\*\*\*\*

Save password

Check Authentication

▸ SASL Settings  
▸ Kerberos Settings

< Back   Next >   Cancel   Finish

# Administrator access to your server's data tree

The screenshot shows a 'New LDAP Connection' dialog box with the following fields and options:

- Authentication Method:** Simple Authentication
- Authentication Parameter:** (empty)
- Bind DN or user:** cn=admin,dc=betrayer,dc=com
- Authorization ID (SASL):** SASL PLAIN only
- Bind password:** (masked with asterisks)
- Save password
- SASL Settings** (expanded)
- Kerberos Settings** (expanded)

A 'Check Authentication' pop-up dialog box is overlaid on top, displaying the message: 'The authentication was successful.' with an 'OK' button.

At the bottom of the main dialog, there are navigation buttons: '< Back', 'Next >', 'Cancel', and 'Finish'. A 'Check Authentication' button is also present near the 'Save password' checkbox.

# Administrator access to your server's data tree

New LDAP Connection

**Authentication**  
Please select an authentication method and input authentication data.

Authentication Method: Simple Authentication

Authentication Parameter: Bind DN or user: cn=admin,dc=betrayer,dc=com

Authorization ID (SASL): SASL PLAIN only

Bind password: \*\*\*\*\*

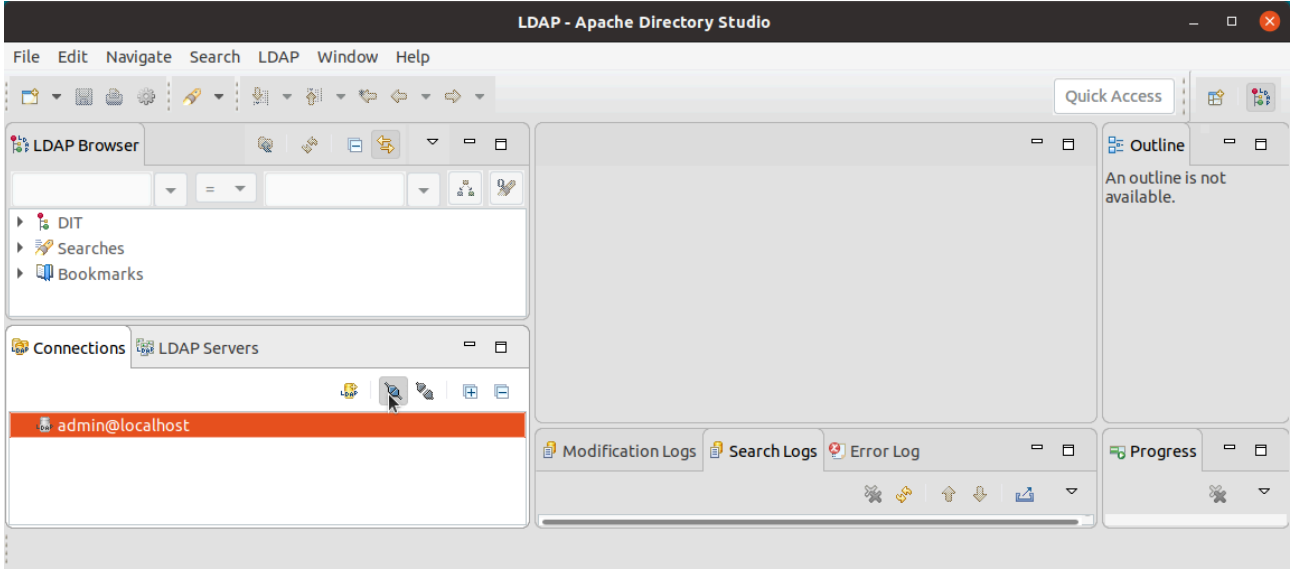
Save password

Check Authentication

▸ SASL Settings  
▸ Kerberos Settings

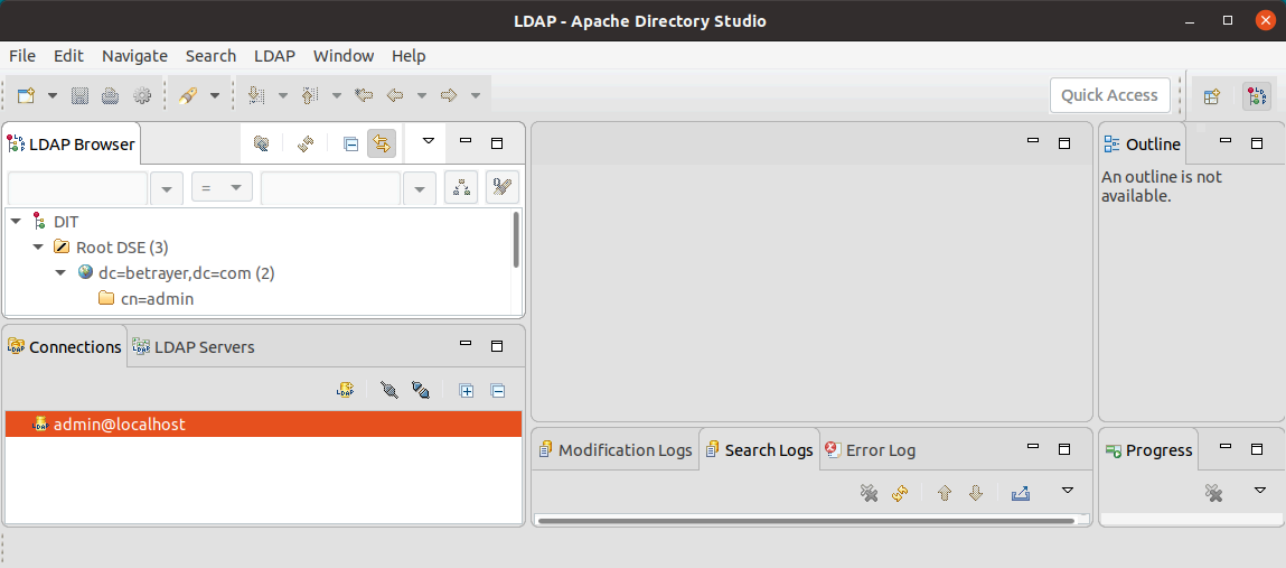
< Back   Next >   Cancel   Finish

# Administrator access to your server's data tree

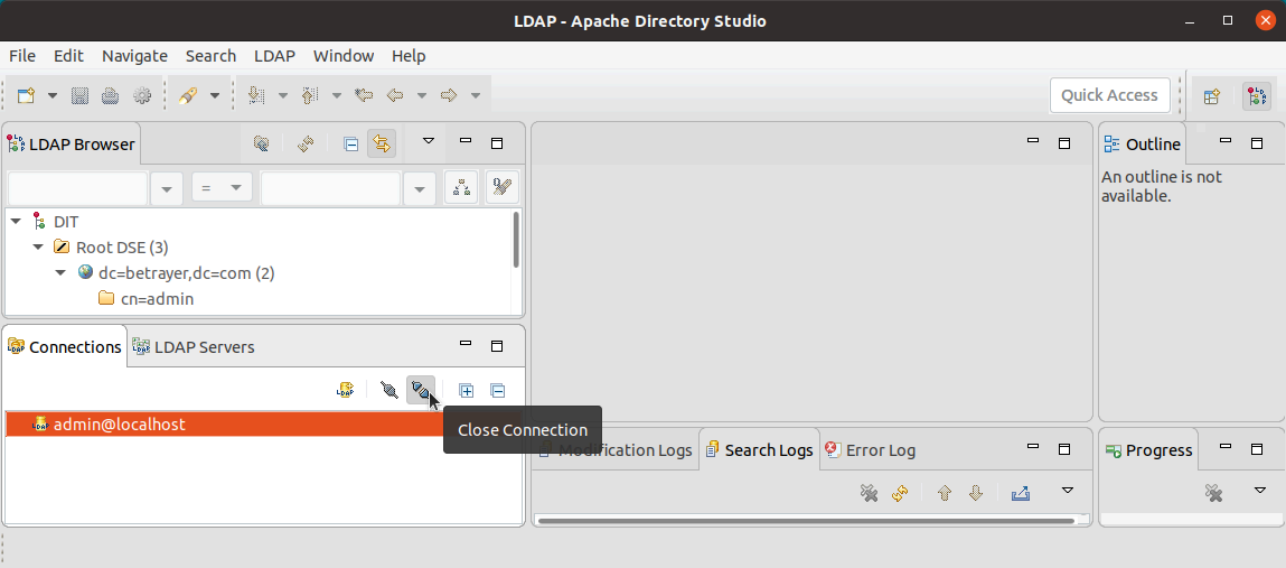




# Administrator access to your server's data tree



# Administrator access to your server's data tree



# Administrator access to your server's configuration

New LDAP Connection

**Network Parameter**  
Please enter connection name and network parameters.

Connection name: admin@config@localhost

Network Parameter

Hostname: localhost

Port: 389

Connection timeout (s): 30

Encryption method: No encryption

Server certificates for LDAP connections can be managed in the ['Certificate Validation'](#) preference page.

Provider: Apache Directory LDAP Client API

Check Network Parameter

Read-Only (prevents any add, delete, modify or rename operation)

< Back Next > Cancel Finish

# Administrator access to your server's configuration

The screenshot shows a 'New LDAP Connection' wizard window. The 'Authentication' section is active, with 'Simple Authentication' selected. The 'Bind DN or user:' field contains 'cn=admin,cn=config', and the 'Authorization ID (SASL):' field is set to 'SASL PLAIN only'. The 'Bind password:' field is masked with asterisks, and the 'Save password' checkbox is checked. A 'Check Authentication' button is visible in the bottom right of the main window. A modal dialog box titled 'Check Authentication' is centered on the screen, displaying a blue information icon and the message 'The authentication was successful.' with an 'OK' button.

**New LDAP Connection**

**Authentication**  
Please select an authentication method and input authentication data.

Authentication Method: Simple Authentication

Authentication Parameter:

Bind DN or user: cn=admin,cn=config

Authorization ID (SASL): SASL PLAIN only

Bind password: \*\*\*\*\*

Save password

Check Authentication

**SASL Settings**

**Kerberos Settings**

< Back   Next >   Cancel   Finish

# Administrator access to your server's configuration

New LDAP Connection

**Browser Options**  
You can specify additional parameters for browsing the directory.

Get base DN's from Root DSE Fetch Base DN's

Base DN:

**Limits**  
Count Limit:   
Time Limit (s):

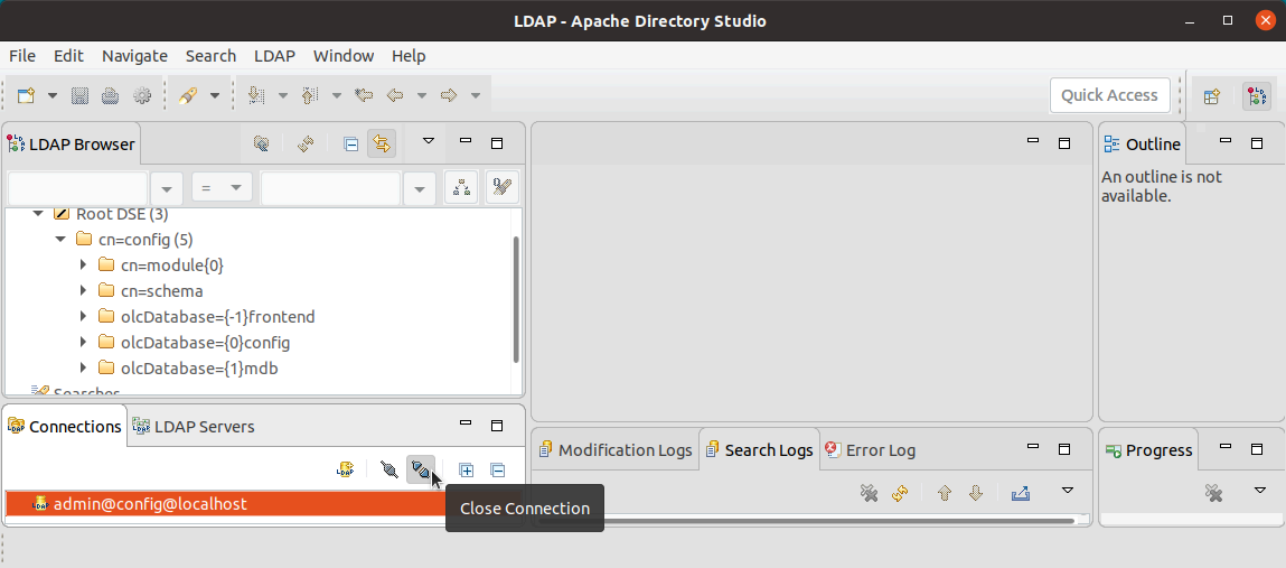
**Aliases Dereferencing**  
 Finding Base DN  
 Search

**Referrals Handling**  
 Follow Referrals manually  
 Follow Referrals automatically  
 Ignore Referrals

**Controls**  
 Use ManageDsaIT control while browsing  
 Fetch subentries while browsing (requires additional search request)  
 Paged Search Page Size:   Scroll Mode

? < Back Next > Cancel Finish

# Administrator access to your server's configuration



# 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

# Adding an entry

---

dn: uid=smith ❶, dc=bet rayer, dc=com ❷  
changetype: add ❸  
objectClass: inetOrgPerson ❹  
objectClass: organizationalPerson ❹  
objectClass: Person ❹  
objectClass: top ❹  
uid: smith ❺  
cn: Jill Smith ❻  
sn: Smith ❻



# Adding a new attribute

Operation	Result
dn: uid=smith, dc=betraye, dc=com ❶ changetype: modify ❷ add: <b>description</b> ❸ <b>description</b> : New employee ❹	dn: uid=smith, dc=betraye, dc=com objectClass: inetOrgPerson objectClass: organizationalPerson objectClass: person objectClass: top cn: Jill Smith sn: Smith uid: smith <b>description</b> : New employee

# Replacing an attribute value

Operation	Result
<p>dn: uid=smith, dc=betraye, dc=com ❶</p> <p>changetype: modify ❷</p> <p>replace: <b>description</b> ❸</p> <p><b>description</b>: Long term employee ❹</p>	<p>dn: uid=smith, dc=betraye, dc=com</p> <p>objectClass: inetOrgPerson</p> <p>objectClass: organizationalPerson</p> <p>objectClass: person</p> <p>objectClass: top</p> <p>cn: Jill Smith</p> <p>sn: Smith</p> <p>uid: smith</p> <p><b>description</b>: Long term employee</p>

# Deleting an attribute entirely

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

## Multi valued attributes

Operation	Result
dn: uid=smith, dc=bet rayer, dc=com changetype: modify add: mail ① mail: smith@company. com ② mail: jsmith@privat eaccount. org ③	dn: uid=smith, dc=bet rayer, dc=com ... sn: Smith mail: jsmith@privat eaccount. org ② mail: smith@company. com ③

# Set semantics of multivalued attributes

Base state	Operation
<p>dn: uid=smith, dc=betray, dc=com objectClass: inetOrgPerson objectClass: organizationalPerson objectClass: person objectClass: top uid: smith commonName: Jill Smith surname: Smith</p>	<p>dn: uid=smith, dc=betray, dc=com changetype: modify add: mail ❶ mail: smith@company.com ❷ mail: jsmith@privateaccount.org ❸ mail: smith@company.com ❹</p> <p>ERR_13207_VALUE_ALREADY_EXISTS ❺ The value 'smith@company.com' already exists in the attribute (mail)</p>

## Deleting selected attribute values

Base state	Operation	Result
dn: uid=smith, dc=betraye, 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=betraye, dc=com ... cn: Jill Smith mail: jsmith@privateaccount.org

# Query scope

---

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

# Query filter

---

- Presence: ( cn=\* )
- Equality: ( ui d=xy123 )
- Comparison: ( 1000 < ui dNumber )
- Substring: ( sur name=K\* )
- Approximate Match: ( sn~gok )

(matches Gack, Keck, Kubi ac, Goi k, Kabbeck, Nguyen Quoc, Gubi c , Koc)

- AND: ( & ( 5 < ui dNumber ) ( sn=\*k\* ) ( gi dNumber=1000 ) )
- OR: ( | ( 5 < ui dNumber ) ( sn=\*k\* ) ( gi dNumber=1000 ) )
- NOT: ( ! ( 5 < ui dNumber ) )
- Extensible match filters, see Extensible Match Search Filter as well.



# 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)

# Implementations

---

access to attrs=matrikelNr

by dn="uid=goik,ou=userlist,dc=hdmstuttgart,dc=de" read

by dn="uid=kuhn,ou=userlist,dc=hdmstuttgart,dc=de" read

by self read

by \* none

access to attrs=userPassword,shadowLastChange,passwordClear

by dn="uid=Administrator,ou=people,ou=M,ou=domainlist,dc=hdmstuttgart,dc=de" read

by anonymous auth

by \* none

# Implementations

---

- JNDI
- Ldapive

# 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=MB,ou=M,dc=beta,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:

- a. Log in to your server's configuration using `cn=admin,cn=config` as in Figure 19.9, “Administrator access to your server's configuration”.
- b. Select your database backend node below `cn=config`.
- c. 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
```

2. Find your LDAP servers database back end. Install the `ldb-utils` package and dump your back end's data.
3. Create an application reading `ldap.h` and `ldmstuttgart.de`'s data and writing corresponding student entries back to your local LDAP server.

# Why MongoDB?

---

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

# Running a Docker container

---

```
docker run -d \  
  - name local MongoDB 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

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

docker-compose up --build -d

d/mongo-init.js:ro

## Manual user creation (mongo-init.js fail)

---

```
> 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"),
```

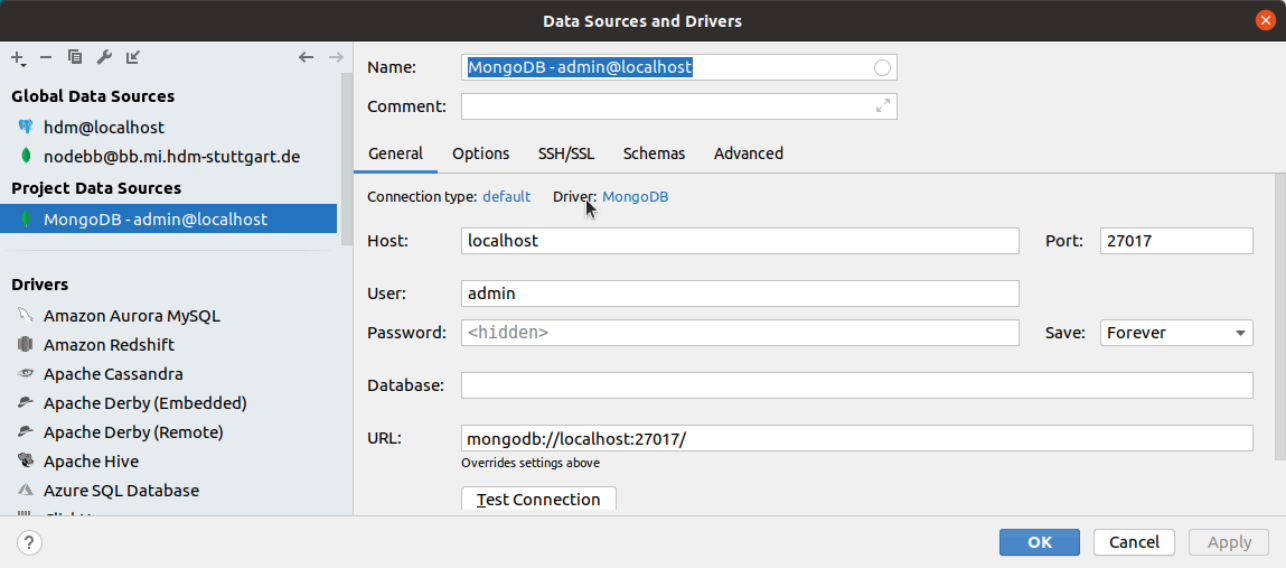
```
  "cname" : "Eve Gardener", "uid" : "gardener", "email" : "gardener@betrayer.com" }
```

# Using IntelliJ

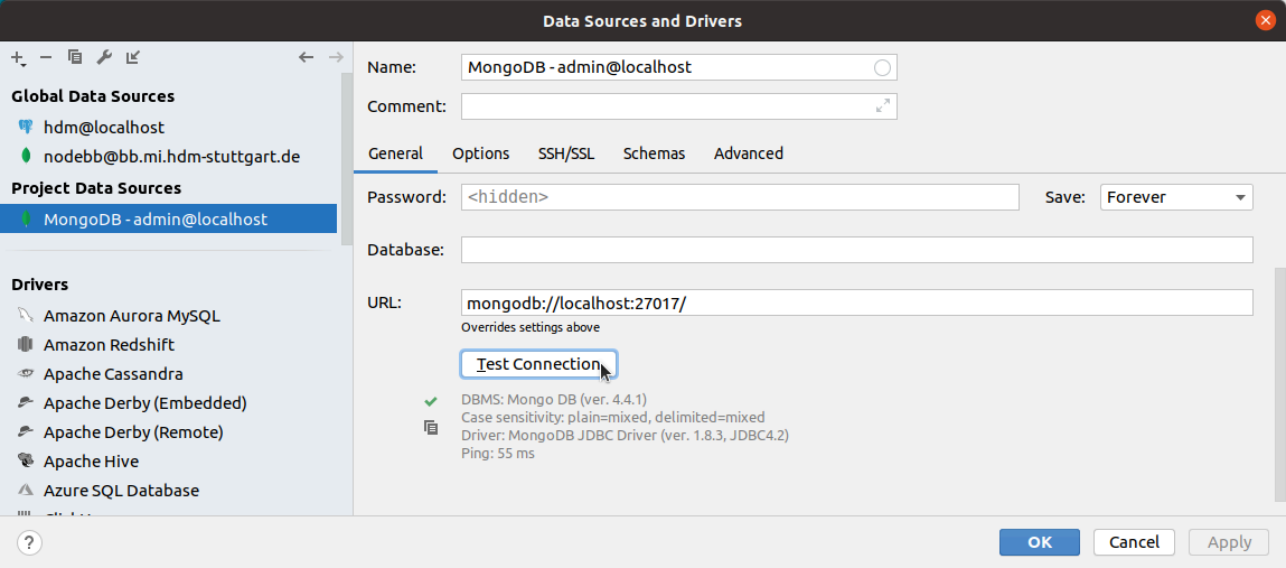
---

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

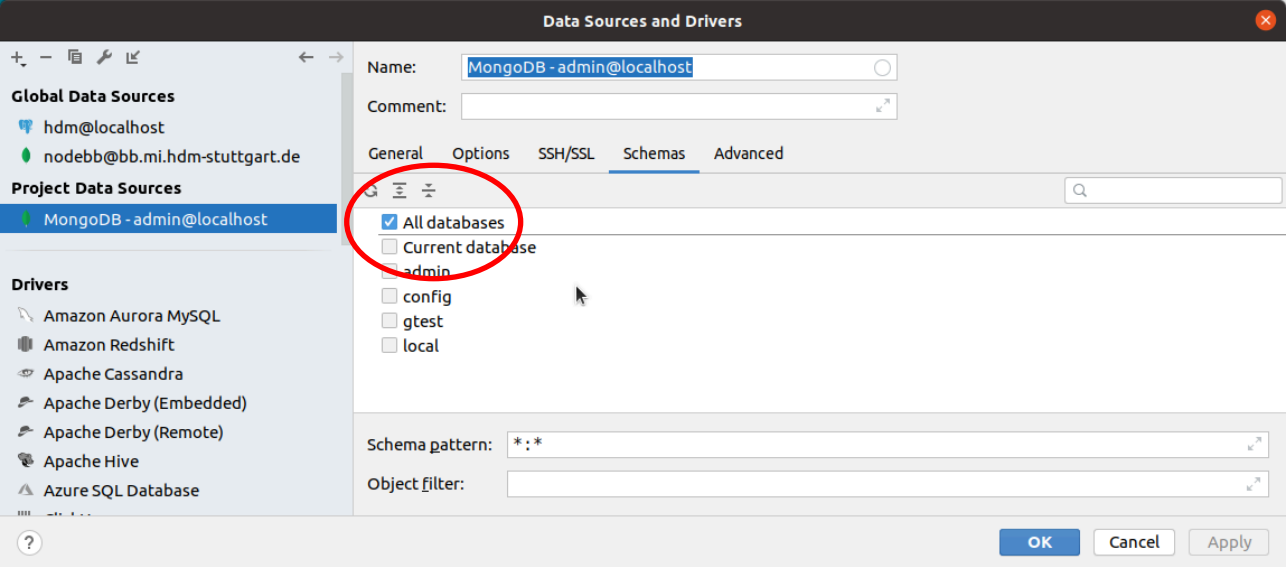
# Idea show all databases



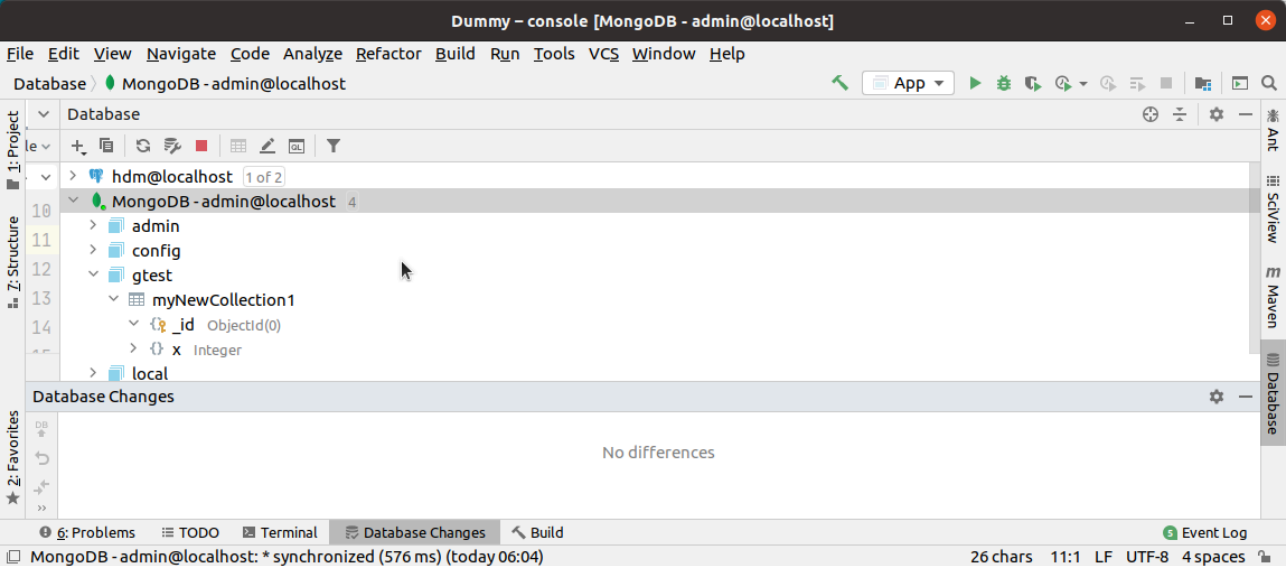
# Idea show all databases



# Idea show all databases



# Idea show all databases



# Terminology / Hierarchy

- 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@betray.com"  
  }  
)
```

## Adding a document

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



# Updating attributes

Code	Result
<pre>db. group. update(   {_id: ObjectId(     "5f a3035932b87a0c60a6ed1a")},   { \$set:     {       cname: "New users" ❶,       groupIdNumber: 1000 ❷     }   } )</pre>	<pre>db. group. find()  [   {     "_id": { "Soi d":       "5f a3035932b87a0c60a6ed1a"},     "cname": "New users" ❶,     "gid": "users",     "gidNumber": 1000 ❷   } ]</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(     "5f a3035932b87a0c60a6ed1a") },   { \$unset:     {       groupIdNumber: 42     }   } )</pre>	<pre>db. group. find()  [   {     "_id": { "Soi d":       "5f a3035932b87a0c60a6ed1a"},     "cname": "My users",     "gi d": "users"   } ]</pre>

# Query filter

---



## 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"
})
```

## 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.

# Implementations

---

# Implementations

---

- 

-

# 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.

# Exercises

---

1.

2.