

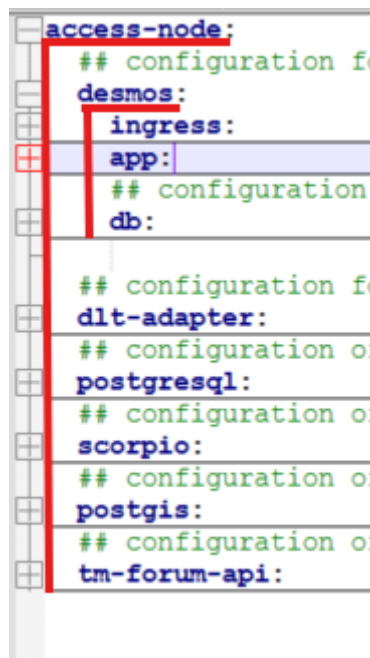
6. AN Configuration: Values.yaml

First, we need to consider whether we've decided to work with secrets (remember rename the file to values.yaml), or not, as the file format may vary slightly.

[Values with secrets.yaml](#)

[Values without secrets.yaml](#)

First of all, we will check that the file header starts as shown in the image:



Now let's configure all the blocks.

Ingress

Modify the ingress hosts and ensure your domain is properly exposed.

```
desmos:
  ingress:
    enabled: true
    className: nginx
    annotations:
      cert-manager.io/cluster-issuer: letsencrypt-in2-test-issuer
    tls:
      - hosts:
          - <YOUR DESMOS DOMAIN>
        secretName: desmos-tls-secret
  app:
```

For example this is our domain: ***desmos.dome-marketplace-lcl.org***

App

The next thing to check is that we have specified your DID key in the following field:

```
app:
  ## profile (test == dev)
  profile: dev
  logLevel:
    root: INFO
    app: DEBUG
  # internal server port
  internalServerPort: 8080
  ## information about the access-node operator
  operator:
    # -- did of the organization running the node
    organizationIdentifier: did:key:XXXXXX
  ## connection information for the context broker
  broker:
```

Next, we will need to review the configured domains:

```
# -- operator external domain
externalDomain:
  protocol: "<YOUR DESMOS DOMAIN PROTOCOL, https RECOMMENDED>"
  domain: "<YOUR DESMOS DOMAIN>"
# -- configuration to set your private key
```

We will need to configure your protocol (in this case it was http but usually it would be https) and the domain of our machine, the URL with which it will be identified.

The next step is to configure the private key, which, if we are working with secrets, will look like this:

```
existingSecret:
  # -- should an existing secret be used
  enabled: true
  # -- name of the secret
  name: access-node-secret
  # -- key to retrieve the password from
  key: desmos-private-key
```

Or without secrets:

```
# -- configuration to set your private key
privateKey:
  value: <YOUR PRIVATE KEY>
learCredentialMachineInBase64:
  value: <YOUR LEAR CREDENTIAL MACHINE IN BASE64>
```

DB

The next step will be to configure our databases.

```
## configuration of the database to be used by the
db:
  # -- existing secret to retrieve the db password
  existingSecret:
    # -- should an existing secret be used
    enabled: true
    # -- name of the secret
    name: access-node-secret
    # -- key to retrieve the password from
    key: desmos-db-password
```

Here we need to configure our database host; this example uses an embedded database. We'll also need to specify our database details and whether or not we're using secrets.

```
## configuration of the database to be us
db:
  password: <YOUR DESMOS DB PASSWORD>
```

As you can see, without secrets, we will need to specify our database password and disable secrets.

DLT-adapter

The next block is the DLT block; here we will need to configure the data obtained with the dome key generator:

```

## configuration for the dlt-adapter - see https://
dlt-adapter:
  existingSecret:
    enabled: true
    name: access-node-secret
    key: dlt-private-key
  env:
    ISS: "<YOUR DLT ADAPTER ISS>"

```



In case of working without secrets:

```

## configuration for the dlt-adapter - s
dlt-adapter:
  env:
    PRIVATE_KEY: <YOUR PRIVATE KEY>
    ISS: "<YOUR DLT ADAPTER ISS>"

```



Postgresql

For the configuration of our Postgres, we must take into account the configuration of the secrets.

```

## configuration of postgres to be used for the blockchain-connector
postgresql:
  auth:
    # -- existing secret
    existingSecret: access-node-secret
    secretKeys:
      adminPasswordKey: desmos-db-password
      userPasswordKey: desmos-db-password

```



And for the case of no secrets:

```

## configuration of postgres to be used for t
postgresql:
  auth:
    password: <YOUR DESMOS DB PASSWORD>

```

Scorpio

For the Scorpio configuration:

```

## configuration of the context-broker - see https://git
scorio:
  ## configuration of the database to be used by broker
  db:
    # -- existing secret to retrieve the db password
    existingSecret:
      # -- should an existing secret be used
      enabled: true
      # -- name of the secret
      name: access-node-secret
      # -- key to retrieve the password from
      key: scorio-db-password

```

And for the configuration without secrets, we would only have to configure the password as shown in the image:

```

## configuration of the context-broker - see htt
scorio:
  ## configuration of the database to be used by
  db:
    password: <YOUR SCORPIO DB PASSWORD>

```

Postgis

```

## configuration of postgis to be used for scorio - see https://git
postgis:
  ## auth configuration for the database
  auth:
    # -- existing secret
    existingSecret: access-node-secret
    secretKeys:
      adminPasswordKey: scorio-db-password
      userPasswordKey: scorio-db-password

```

And without secrets:

```

## configuration of postgis to be used for sc
postgis:
  ## auth configuration for the database
  auth:
    password: <YOUR SCORPIO DB PASSWORD>

```

This would be all the configuration to take into account to prepare our AN to be able to perform the synchronization tests.

Logically, there are more blocks in the file that we haven't mentioned, but we suggest not editing them without having the necessary knowledge.

Revision #4

Created 24 November 2025 07:57:02 by Miguel Mir

Updated 26 November 2025 17:19:58 by Miguel Mir