

## Operating Guide on Databricks Platform Infrastructure

<b>Document Owner(s)</b>	@Kevin Yang @Jay Jiang
<b>Work Package</b>	WP4
<b>Status</b>	UNDER REVIEW
<b>Consulted</b>	
<b>List of Approvers</b>	

### Table of Contents




Scope and Source code Repository

Databricks Workspaces

Non-prod Workspaces

Prod workspaces

Steps to interact with Workspaces

- 1 Confirm or change the Ruleset in for network security-group
- 2 Interacting and provisioning with AWS Infrastructure
- 3 Provision Databricks Account-level Resources (Cloud Resources)
- 4 Provision Databricks Account-level Resources (BU Resources)  
 Gitlab Variable provisioning
- 5 Remaining Databricks Workspace-level Resources
- 6 Populate Allowed JARs/Init Scripts to Workspace Catalogs  
 Logging into Databricks workspaces  
 Examining available Catalogs & associated backend for the given workspace

### Scope and Source code Repository

The [IaC \(Infrastructure-as-Code\)](#) codebases consist of the five repositories to provision AWS resources, Databricks Account Resources, and Databricks workspace resources used to power Datahub-databricks platform.

In the multi-step deployment processes, the subsequent repositories may have Terraform Remote State dependencies link to its previous deployment, more details on Git Repo can be found in here

[Git Repo Topology - Data Platform Repos - Future Networks - Data Hub and Analytics Project - Confluence \(atlassian.net\)](#)

The Gitlab repository can be also summarized in the table below:

Repository name	Purposes
1. <a href="#">app-datahub-nonprod-databricks-aws-infra</a>	Underlying Databricks Infrastructure for <b>Non-prod</b> from <code>aws</code> Terraform provider, except VPC (Virtual Private Cloud Component) resources
2. <a href="#">app-datahub-prod-databricks-aws-infra</a>	Underlying Databricks Infrastructure for <b>Prod</b> from <code>aws</code> Terraform provider, except VPC (Virtual Private Cloud Component) resources following the approval based on the <b>Non-prod</b> environment
3. <a href="#">core-network-databricks-vpc-components</a>	data block for VPC and Subnets with <code>security-group</code> and <code>private-link-endpoints</code> resources for both <b>Non-prod</b> and <b>Prod</b>
4. <a href="#">databricks-unity-catalog</a>	Account-level (and workspace-agnostic) resources for <code>databricks</code> Terraform provider for both <b>prod</b> and <b>nonprod</b> ,  <i>incl. resources related for Databricks permission grants which are <a href="#">workspace agnostic despite leveraging workspace-level APIs</a>.</i>
5. <a href="#">databricks-workspaces</a>	Workspace-level for <code>databricks</code> workspace provider, namely <code>&lt;Business Unit&gt;_lab</code> or <code>&lt;Business Unit&gt;_field</code> workspaces

with the corresponding workspace ID specified via the `host` parameter.

## Databricks Workspaces

The following table summarise the workspaces provisioned within Databricks platform according to the following solution design [Infrastructure Design - Future Networks - Data Hub and Analytics Project - Confluence \(atlassian.net\)](#).

### Non-prod Workspaces

Workspace Names	Link	Purpose
<b>Non-prod:</b> digital-lab- workspace- nonprod	<a href="https://dbc-eaba2339-eb1e.cloud.databricks.com/">https://dbc-eaba2339-eb1e.cloud.databricks.com/</a>	Databricks workspaces for Digital BU in <b>LAB</b> lab environment
<b>Non-prod:</b> digital-field- workspace- nonprod	<a href="https://dbc-fecbb5ff-7592.cloud.databricks.com/">https://dbc-fecbb5ff-7592.cloud.databricks.com/</a>	Databricks workspaces for Digital BU in <b>FIELD</b> environment
<b>Non-prod:</b> elec-network-lab- workspace- nonprod	<a href="https://dbc-de6c0ca1-0e35.cloud.databricks.com/">https://dbc-de6c0ca1-0e35.cloud.databricks.com/</a>	Databricks workspaces for Electricity Network BU in <b>LAB</b> environment
<b>Non-prod:</b> elec-network-field- workspace- nonprod	<a href="https://dbc-94129c9d-8f32.cloud.databricks.com/">https://dbc-94129c9d-8f32.cloud.databricks.com/</a>	Databricks workspaces for Electricity Network BU in <b>FIELD</b> environment

### Prod workspaces

Workspace Names	Link	Purpose
Prod: digital-lab-workspace-prod	<a href="https://jemena-digital-lab.cloud.databricks.com/">https://jemena-digital-lab.cloud.databricks.com/</a>	Databricks workspaces for Digital BU in LAB lab environment
Prod: digital-field-workspace-prod	<a href="https://jemena-digital-field.cloud.databricks.com/">https://jemena-digital-field.cloud.databricks.com/</a>	Databricks workspaces for Digital BU in FIELD environment
Prod: elec-network-lab-workspace-prod	<a href="https://jemena-elec-network-lab.cloud.databricks.com/">https://jemena-elec-network-lab.cloud.databricks.com/</a>	Databricks workspaces for Electricity Network BU in LAB environment
Prod: elec-network-field-workspace-prod	<a href="https://jemena-elec-network-field.cloud.databricks.com/">https://jemena-elec-network-field.cloud.databricks.com/</a>	Databricks workspaces for Electricity Network BU in FIELD environment

## Steps to interact with Workspaces

### 1 Confirm or change the Ruleset in for network security-group

As first point of interactions, navigate and clone [Jemena / Projects / Future Networks Datahub / Databricks / core-network-databricks-vpc-components · GitLab](#) to your IDE, you would see the repository structure exhibits the following, where it contains both

- the **security-groups** configured for ingress/egress for traffics for Databricks connectivity between the **control plane** and the **data plane**
- the **private-link** resources used to deploy Private VPC endpoint as per security compliance for private connectivity

### i Databricks operates out of a **control plane** and a **compute plane**.

- The **control plane** includes the backend services that Databricks manages in your Databricks account. The web application is in the control plane.

- The **compute plane** is where your data is processed. There are two types of compute planes depending on the compute that you are using.

```
1 |— README.md
2 |— backend.tf
3 |— main-frontend-privatelink.tf
4 |— main-npd-vpc-resources.tf
5 |— main-prod-vpc-resources.tf
6 |— modules
7 |   |— private-link-endpoints
8 |       |— README.md
9 |       |— main.tf
10 |       |— outputs.tf
11 |       |— variables.tf
12 |       |— versions.tf
13 |   |— security-group
14 |       |— README.md
15 |       |— main.tf
16 |       |— outputs.tf
17 |       |— variables.tf
18 |       |— versions.tf
19 |— output.tf
20 |— providers.tf
21 |— shared-locals.tf
22 |— shared.auto.tfvars
23 |— variables.tf
```

⚠ Please make changes via a **feature branch** before submitting a merge request to main branch. CI/CD is configured to perform

- **terraform plan** on feature branch
- and both **terraform plan** and **terraform apply** on the main branch

**navigate to** `./shared_locals.tf`, and *append the ingress/egress rules to the existing* `data.aws_subnet`, by declaring a variable in the `locals` block, in the syntax of `allow<port>`

```
1 | # Allowing 2443 bidirectional traffic
2 | allow2443 = [for cidr in values(data.aws_subnet.nonprod_subnets).*.cidr_block :
3 |   {
4 |     rule_description = "Allow cluster bidirectional connectivity on 2443"
5 |     ip_protocol      = "tcp"
6 |     from_port        = 2443
7 |     to_port          = 2443
8 |     cidr_ipv4         = "${cidr}"
9 |   }
10 | ]
```


then, navigate to `<nonprod/prod>_vpc_resources.tf` for `nonprod` or `prod_vpc_resources.tf` for `prod`, and append the port to your `nonprod_dataplane_sg_config` variable nested in `locals` block.

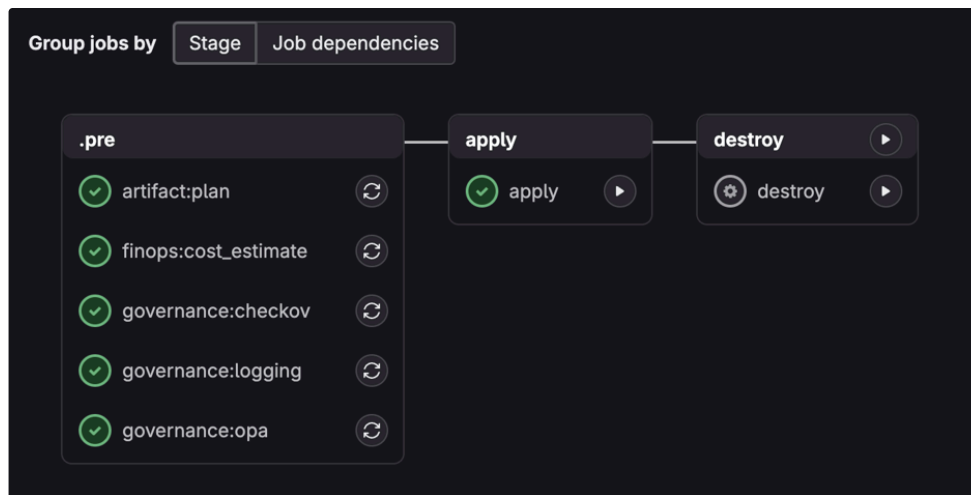
For the example below, ruleset such as `local.allow443`, `local.allow6666`, `local.allow2443` has been appended into the terraform scripts, for bidirectional traffic as per defined for [backend private-link connectivity \(Databricks Official Doc\)](#). You may customize single direction via appending only to `egress` and `ingress` rules.

```
1  nonprod_dataplane_sg_config = {
2    "dataplane-tcp-private-links" = {
3      create_sg    = true
4      name         = "${var.nonprod_prefix}-sg-dataplane-tcp-bidirectional"
5      description  = "Common bidirectional security group for databricks nonprod VPC private
links"
6      egress       = concat(local.allow443, local.allow6666, local.allow2443)
7      ingress      = concat(local.allow443, local.allow6666, local.allow2443)
8    }
9  }
```

on the same directory, comment out the `nonprod_private_link_endpoints` and/or `prod_private_link_endpoints`, as we will want to provision ENI after configuring network traffic:

```
1  # Comment out the following
2  module "nonprod_private_link_endpoints" {
3    depends_on    = [module.nonprod_databricks_common_workspace_dataplane_sg]
4    source         = "./modules/private-link-endpoints/"
5    prefix        = var.nonprod_prefix
6    vpc_id        = var.core_network_shared_nonprod_vpc_id
7    eni_subnet_names = var.nonprod_eni_subnet_names
8    security_group_ids = [
9      module.nonprod_databricks_common_workspace_dataplane_sg.security_group_ids.dataplane-
tcp-private-links
10   ]
11    tags = local.nonprod_additional_tags
12  }
```

apply the resources onto the AWS platform after GitLab gitlab merge request, where the CI/CD pipeline would've been setout to perform the following stages (with apply that can be manually triggered via  button shown below).



Upon completion, `security-group` s and for `prod/non-prod` would've been completed.

navigating back to `<nonprod/prod>_vpc_resources.tf` , and comment the following:

```

1  # Uncomment the following
2  module "nonprod_private_link_endpoints" {
3      depends_on      = [module.nonprod_databricks_common_workspace_dataplane_sg]
4      source           = "./modules/private-link-endpoints/"
5      prefix          = var.nonprod_prefix
6      vpc_id           = var.core_network_shared_nonprod_vpc_id
7      eni_subnet_names = var.nonprod_eni_subnet_names
8      security_group_ids = [
9          module.nonprod_databricks_common_workspace_dataplane_sg.security_group_ids.dataplane-
10 tcp-private-links
11 ]
12 tags = local.nonprod_additional_tags
13 }

```

and follow the plan/apply steps above, `private-link-endpoints` will be up.

## 2 Interacting and provisioning with AWS Infrastructure

Depending on your AWS environment (Prod/Non-prod), navigate to either

- **Non-prod:** [Jemena / Projects / Future Networks Datahub / Databricks / app-datahub-nonprod-databricks-aws-infra · GitLab](#)
- **Prod:** [Jemena / Projects / Future Networks Datahub / Databricks / app-datahub-prod-databricks-aws-infra · GitLab](#)

where the directory is structured ( `nonprod` as example) as follows:

View databricks-aws-infra structures

- 1 |— README.md
- 2 |— backend.tf

```
3 |─ main-account-data-egress.tf
4 |─ main-account-log-storage.tf
5 |─ main-account-metastore-storage.tf
6 |─ main-account-metastore-uc-role.tf
7 |─ main-digital-bu-catalog-cloud-resources.tf
8 |─ main-digital-bu-wks-cloud-resources.tf
9 |─ main-elec-network-bu-catalog-cloud-resources.tf
10 |─ main-elec-network-bu-wks-cloud-resources.tf
11 |─ main-secrets.tf
12 |─ modules
13 |   |─ bucket
14 |     |─ README.md
15 |     |─ default_policies.tf
16 |     |─ locals.tf
17 |     |─ main.tf
18 |     |─ outputs.tf
19 |     |─ variables.tf
20 |   |─ catalog-cloud-resources
21 |     |─ catalog_buckets.tf
22 |     |─ ext_tbl_access_role.tf
23 |     |─ locals.tf
24 |     |─ output.tf
25 |     |─ uc_data_access_role.tf
26 |     |─ variables.tf
27 |   |─ metastore-iam-role
28 |     |─ main.tf
29 |     |─ outputs.tf
30 |     |─ variables.tf
31 |   |─ secrets
32 |     |─ main.tf
33 |     |─ outputs.tf
34 |     |─ variables.tf
35 |   |─ security-group
36 |     |─ README.md
37 |     |─ main.tf
38 |     |─ outputs.tf
39 |     |─ variables.tf
40 |     |─ versions.tf
41 |   |─ workspace-cloud-resources
42 |     |─ README.md
43 |     |─ bucket-policies.tf
44 |     |─ buckets.tf
45 |     |─ cross-account-medallion-pass-role.tf.skip
46 |     |─ cross-account-role.tf
47 |     |─ locals.tf
48 |     |─ outputs.tf
49 |     |─ security-groups.tf
50 |     |─ variables.tf
51 |─ (nonprod/nonprod).auto.tfvars
52 |─ outputs.tf
53 |─ providers.tf
54 |─ shared-data.tf
55 |─ shared-locals.tf
56 |─ tags.tf
57 |─ variables.tf
```

For creating the BU resources, first navigate to or create `main-<Business Unit>-bu-catalog-cloud-resources.tf`, and define the syntax like the following, and substituting the `<>` with the related parameters:

```
1 module "digital_bu_catalog_cloud_resources" {
2   for_each = toset(["lab", "qa", "field"])
3   source   = "../modules/catalog-cloud-resources"
4
5   environment          = var.environment
6   aws_account_id       = var.aws_account_id
7   aws_region           = "ap-southeast-2"
8   databricks_account_id = var.databricks_account_id
9   stage_storage_credential_external_id = var.databricks_account_id //placeholder
10  databricks_control_plane_aws_account_id = var.databricks_control_plane_aws_account_id
11  bu_kms_key_arn        = var.digital_kms_key_arn
12  business_unit         = "digital"
13  bucket_name_prefix    = "${var.prefix}-${var.environment}"
14  data_stage            = each.value
15 }
16
17
```

Subsequently, navigate to or create `main-<Business Unit>-bu-wks-cloud-resources.tf`, and edit the following:

```
1 locals {
2   <Business Unit>_bu_lab_workspace_name = "<Business Unit>-lab-
workspace-${var.environment}"
3   <Business Unit>_bu_field_workspace_name = "<Business Unit>-field-
workspace-${var.environment}"
4
5   <Business Unit>_bu_lab_workspace_buckets = {
6     "root" = "${var.prefix}-s3-${local.digital_bu_lab_workspace_name}-root"
7   }
8   <Business Unit>_bu_field_workspace_buckets = {
9     "root" = "${var.prefix}-s3-${local.digital_bu_field_workspace_name}-root"
10  }
11 }
```

and assigning the following variable at `(nonprod/prod).auto.tfvars` :

**i** Note that the underlying `aws_account_id` (prod or non-prod) is a member account that sits on a larger Organizational Units (OU, i.e. `core_network_aws_account_id`). Refer to this [AWS documentation](#) for more info.

The resources coming from OU and requires platform team's implementation incl. but not limited to:

- `metastore_kms_key`
- `private_eni_subnet_names`
- `<Business Unit>_kms_key_arn`

```

1 environment                                = <prod/nonprod>
2 target_vpc_id                             = ""
3 databricks_account_id                     = ""
4 databricks_control_plane_aws_account_id   = ""
5 aws_account_id                            = ""
6 core_network_aws_account_id               = ""
7
8 #kms_alias: arn:aws:kms:ap-southeast-2:<account ID>:alias/app-datahub-<nonprod/prod>-
  databricks-metastore-kms-key
9 metastore_kms_key_arn = "arn:aws:kms:ap-southeast-2:<account ID>:key/f8cee53a-4749-4935-
  9f17-fcf#####"
10 private_eni_subnet_names = [
11     "app-datahub-dev-eni-az<#>-<#1>" # e.g. "app-datahub-dev-eni-az1-01"
12     , "app-datahub-dev-eni-az<#>-<#2>"
13     , "app-datahub-dev-eni-az<#>-<#3>"
14 ]
15
16 #app-datahub-<nonprod/prod>-databricks-digital-kms-key
17 digital_kms_key_arn = "arn:aws:kms:ap-southeast-2:<account ID>:key/e0#####-b####-####-####-
  #####"
18 #app-datahub-<nonprod/prod>-databricks-elec-network-kms-key
19 elec_network_kms_key_arn = "arn:aws:kms:ap-southeast-2:<account ID>:key/61f#####-b####-####-
  ####-#####"
20
21 #app-datahub-<nonprod/prod>-databricks-<Business Unit>-kms-key
22 ...
23

```

Whilst provisioning the cloud resources, referred to the pipeline below to identify required edits to pass the plug-in, e.g. `checkov` , `opa` , and `logging` .

**⚠ DO NOT** trigger **destroy** as it may have impact to the subsequent provisioning

**For** `checkov` [warnings](#) related to legacy Terraform version, add the following comment format within the resource block to resolve specific warning number i.e.

`CKV2_AWS_<#>`

*For example, for disabling S3 bucket warning due to standalone public access or lifecycle configuration not been configure as S3 attribute (Optional, **Deprecated**):*

```

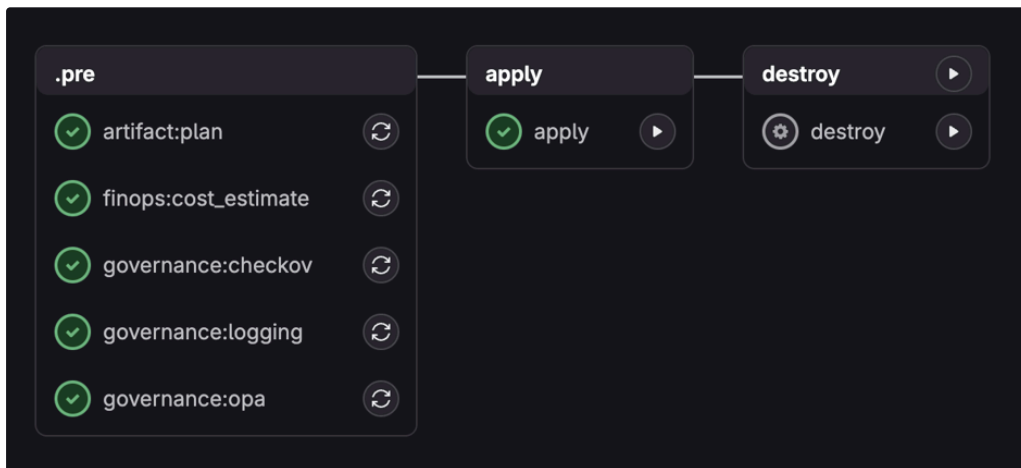
1 resource "aws_s3_bucket" "this" {
2     #checkov:skip=CKV2_AWS_61: lifecycle configuration declared in the resource block
  below

```

```

3   #checkov:skip=CKV2_AWS_6: S3 bucket Public Access Block declared in the resource
    block below
4
5   ...
6 }

```



Cloud resource CI/CD workflows

Lastly, click through the **apply** text to view the outputs, take note of the structure as you may need to pass your terraform state files, with example below:

```

1 Releasing state lock. This may take a few moments...
2 Apply complete! Resources: 0 added, 6 changed, 0 destroyed.
3 Outputs:
4 digital_bu_catalog_cloud_resources = {
5   "field_catalog" = {
6     "bronze" = {
7       "arn" = "arn:aws:s3:::app-datahub-prod-s3-digital-field-bronze"
8       "name" = "app-datahub-nonprod-s3-digital-field-bronze"
9     }
10    "data_access_iam_role" = {
11      "arn" = "arn:aws:iam::<account-ID>:role/digital-field-ctlg-uc-data-access"
12      "name" = "digital-field-ctlg-uc-data-access"
13    }
14  }
15  ...

```

### 3 Provision Databricks Account-level Resources (Cloud Resources)

Following provisioning on AWS provider, head to [Jemena / Projects / Future Networks Datahub / Databricks / databricks-unity-catalog](#) to further provisioning resources.

To kickstart Databricks platform, examine the following repository structure (dropdown):

View databricks-unity-catalog structure

```

1 |— README.md
2 |— modules

```

```
3 |   └─ log-delivery
4 |     └─ main-log-delivery.tf
5 |     └─ outputs.tf
6 |     └─ providers.tf
7 |     └─ variables.tf
8 |   └─ managed-workspace
9 |     └─ README.md
10 |    └─ outputs.tf
11 |    └─ providers.tf
12 |    └─ variables.tf
13 |    └─ wks_cmk.tf
14 |    └─ wks_config.tf
15 |   └─ name
16 |     └─ README.md
17 |     └─ example
18 |       └─ test_names.tf
19 |     └─ main.tf
20 |   └─ rbac-hierarchy
21 |     └─ README.md
22 |     └─ main-access-groups.tf
23 |     └─ main-functional-groups.tf
24 |     └─ main-workspace-access.tf
25 |     └─ main-workspace-service-principal.tf
26 |     └─ outputs.tf
27 |     └─ providers.tf
28 |     └─ variables.tf
29 |   └─ service-principal-gitlab
30 |     └─ README.md
31 |     └─ data.tf
32 |     └─ main-gitlab-var.tf
33 |     └─ outputs.tf
34 |     └─ providers.tf
35 |     └─ variables.tf
36 | └─ scripts
37 |   └─ metastore-destroy.sh
38 |   └─ tf-apply.sh
39 |   └─ tf-plan.sh
40 | └─ stacks
41 |   └─ cloud-resources
42 |     └─ backend.tf
43 |     └─ main.tf
44 |     └─ outputs.tf
45 |     └─ providers.tf
46 |     └─ upstream_stacks.tf
47 |     └─ variables.tf
48 |   └─ digital-bu
49 |     └─ backend.tf
50 |     └─ dependencies.tf
51 |     └─ locals.tf
52 |     └─ main-bu-cred.tf
53 |     └─ main-catalog-field.tf
54 |     └─ main-catalog-lab.tf
55 |     └─ main-system-schema.tf
56 |     └─ main-workspace-field.tf
57 |     └─ main-workspace-lab.tf
58 |     └─ main-xacc-resources.tf
59 |     └─ outputs.tf
60 |     └─ providers.tf
```

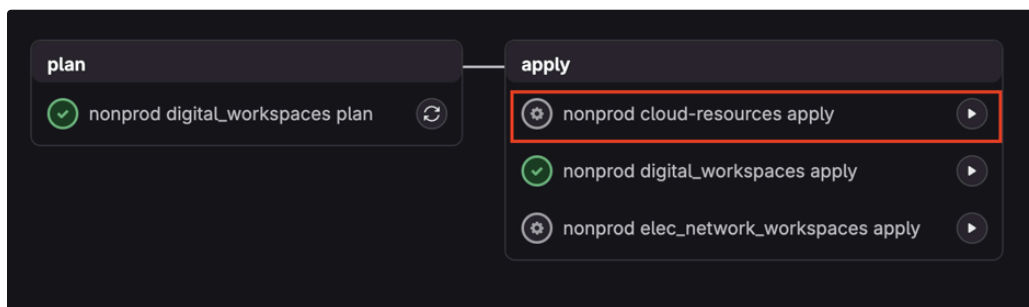
```

61 | | | sim-functional-groups.tf
62 | | | sim-group-membership.tf
63 | | | variables.tf
64 | | └─ elec-network-bu
65 | |   | └─ backend.tf
66 | |   | └─ dependencies.tf
67 | |   | └─ locals.tf
68 | |   | └─ main-bu-cred.tf
69 | |   | └─ main-catalog-field.tf
70 | |   | └─ main-catalog-lab.tf
71 | |   | └─ main-common-grants.tf
72 | |   | └─ main-system-schema.tf
73 | |   | └─ main-workspace-field.tf
74 | |   | └─ main-workspace-lab.tf
75 | |   | └─ outputs.tf
76 | |   | └─ providers.tf
77 | |   | └─ repair.tf
78 | |   | └─ sim-functional-groups.tf
79 | |   | └─ sim-group-membership.tf
80 | |   | └─ variables.tf
81 | | └─ metastore
82 | |   | └─ backend.tf
83 | |   | └─ main.tf
84 | |   | └─ outputs.tf
85 | |   | └─ providers.tf
86 | |   | └─ upstream_stacks.tf
87 | |   | └─ variables.tf

```

⚠ Note that **metastore apply** has been disabled to allow idempotent operations for other account-wide activity. i.e. edit `./stacks/metastore` with cautions

With `rest_backend` and `scc_relay` provisioned via `cloud-resources`, simply re-run cloud resource applies via existing CI/CD setup:



#### 4 Provision Databricks Account-level Resources (BU Resources)

BU workspace provisioning occurs via stacks repository (browse through `digital-bu` and `elec-network-bu` for setup example).

ⓘ NOTE that `grants.tf` and `catalog.tf` and any Terraform blocks associated with

- databricks.workspace
- databricks.lab\_mws
- databricks.field\_mws

will undergo after workspaces resources are provisioned.

Specifically, `./stacks/<BU-stack-folder>/locals.tf` has been setup for the respective business unit workspace that fetches the resources previously provisioned.

For `subnets` ID mapping to workspace and corresponding environment, refer to the following page for more information [Infrastructure Design | VPC & Subnets](#)

```

1  locals {
2    business_unit = <To be filled-in>
3
4    //account level resources
5
6    target_vpc_id = {
7      nonprod = "vpc-0ff11056a20a2ce44"
8      prod    = "vpc-002f643f2c6498c81"
9    }
10
11   scc_eni_id = {
12     nonprod =
13 data.terraform_remote_state.nonprod_mws_endpoints.outputs.scc_relay_endpoint_id
14     prod    = data.terraform_remote_state.prod_mws_endpoints.outputs.scc_relay_endpoint_id
15   }
16
17   rest_eni_id = {
18     nonprod =
19 data.terraform_remote_state.nonprod_mws_endpoints.outputs.rest_backend_endpoint_id
20     prod    =
21 data.terraform_remote_state.prod_mws_endpoints.outputs.rest_backend_endpoint_id
22   }
23
24   metastore_id = {
25     nonprod = data.terraform_remote_state.nonprod_metastore.outputs.metastore_id
26     prod    = data.terraform_remote_state.prod_metastore.outputs.metastore_id
27   }
28
29   metastore_admin_gid = {
30     nonprod = data.terraform_remote_state.nonprod_metastore.outputs.metastore_admin_group_id
31     prod    = data.terraform_remote_state.prod_metastore.outputs.metastore_admin_group_id
32   }
33 }
34
35 locals {
36   // workspace level resources
37   workspace_config = {
38     nonprod = {

```

```

37     digital_catalogs_iam_role_arn =
data.terraform_remote_state.nonprod_aws_infra.outputs.digital_bu_catalog_cloud_resources.lab
_catalog.data_access_iam_role.arn
38     digital_ext_tbl_schema_role_arn =
data.terraform_remote_state.nonprod_aws_infra.outputs.digital_bu_catalog_cloud_resources.lab
_catalog.ext_tbl_data_access_iam_role.arn
39     digital_bu_lab = {
40         subnets      = ["subnet-039d49c832c613766", "subnet-04264bb265a9b477f", "subnet-
06c48e0b4be6a9b1c"] //pre-defined
41         sgs           =
data.terraform_remote_state.nonprod_aws_infra.outputs.digital_bu_lab_wks_cloud_resources["se
curity_group_ids"]
42         root_bucket   =
data.terraform_remote_state.nonprod_aws_infra.outputs.digital_bu_lab_wks_cloud_resources["wo
rkspace_root_bucket_name"]
43         xacc_role     =
data.terraform_remote_state.nonprod_aws_infra.outputs.digital_bu_lab_wks_cloud_resources["cr
oss_account_role_arn"]
44     }
45     digital_bu_field = {
46         subnets      = ["subnet-011451954924d667a", "subnet-057a5a1f9b2d4d983", "subnet-
0d360b9a1fdb82ccb"] //pre-defined
47         sgs           =
data.terraform_remote_state.nonprod_aws_infra.outputs.digital_bu_field_wks_cloud_resources["
security_group_ids"]
48         root_bucket   =
data.terraform_remote_state.nonprod_aws_infra.outputs.digital_bu_field_wks_cloud_resources["
workspace_root_bucket_name"]
49         xacc_role     =
data.terraform_remote_state.nonprod_aws_infra.outputs.digital_bu_field_wks_cloud_resources["
cross_account_role_arn"]
50     }
51 }
52 prod = ...
53 }
54 }

```

Navigate to `.gitlab-ci.yml`, establish BU plan and apply as Terraform state files are separated based on Business Units. With `digital` BU as an example:

```

1  nonprod digital_workspaces plan:
2  stage: plan
3  # artifacts:
4  #   paths:
5  #     - stacks/*/*.tfplan
6  variables:
7      STATEFILE_NAME: "digital-bu"
8      ENV: "nonprod"
9      DATABRICKS_HOST: "https://accounts.cloud.databricks.com"
10     DATABRICKS_CLIENT_ID: $app_datahub_nonprod_databricks_account_sp_clientid
11     DATABRICKS_CLIENT_SECRET: $app_datahub_nonprod_databricks_account_sp_secret
12     DATABRICKS_ACCOUNT_ID: $app_datahub_nonprod_databricks_account_id
13     TF_VAR_business_unit: "digital"
14     TF_VAR_client_id: $DATABRICKS_CLIENT_ID
15     TF_VAR_client_secret: $DATABRICKS_CLIENT_SECRET

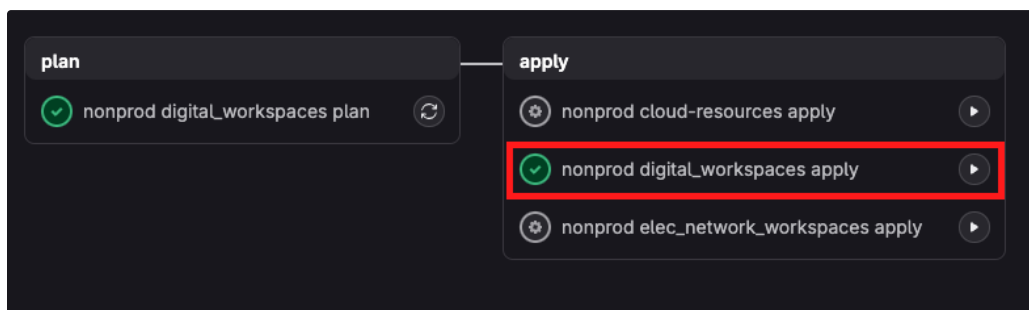
```

```

16 TF_VAR_databricks_account_id: $DATABRICKS_ACCOUNT_ID
17 TF_VAR_aws_account_id: $nonprod_aws_account_id
18 # TF_VAR_target_vpc_id: $nonprod_vpc_id
19 TF_VAR_ci_job_token: $CI_JOB_TOKEN
20 before_script:
21   - *terraform-ver-init
22 script:
23   - cd stacks/digital-bu
24   - ../../scripts/tf-plan.sh
25 rules:
26   - changes:
27     - stacks/digital-bu/**/*
28     - modules/managed-workspace/**/*
29     - modules/rbac-hierarchy/**/*

```

Finally, on Gitlab repository, go to **Operate** (Column bar on the Left-hand side) > Under Business Unit Terraform State (**nonprod-digital-bu** for example) > click through the **Pipeline #** > click the BU text with image shown below.



databricks-unity-catalog pipeline workflow

Record the corresponding

- `workspace_url`,
- `ci_sp_client_id`, and
- depending on *prod* or *nonprod*, either record secret in **nonprod** or conduct data block `terraform_remote_state` call for `<lab/field>_ci_sp_oidc["secret"]` from prod

Establish the remaining `databricks.workspaces` modules by updating BU (in this case `stacks/digital-bu` and `stacks/elec-network-bu`) workspace host URL `providers.tf` below

```

1 locals {
2
3   # updating workspace host URL below
4   workspace_host_url = {
5     nonprod_lab    = "https://dbc-#####-###.cloud.databricks.com"
6     nonprod_field  = "https://dbc-#####-###.cloud.databricks.com"
7     prod_lab       = "https://jemena-digital-lab.cloud.databricks.com"

```

```

8     prod_field    = "https://jemena-digital-field.cloud.databricks.com"
9 }
10 }
11
12 provider "databricks" {
13     alias        = "account"
14     host         = "https://accounts.cloud.databricks.com"
15     account_id   = var.databricks_account_id
16 }
17
18 provider "databricks" {
19     //any workspace provider is fine, this is used for grants, see
    https://registry.terraform.io/providers/databricks/databricks/latest/docs/resources/grant
20     alias = "lab_mws"
21     host  = local.workspace_host_url["${var.environment}_lab"] //digital lab
22 }
23
24 provider "databricks" {
25     //any workspace provider is fine, this is used for grants, see
    https://registry.terraform.io/providers/databricks/databricks/latest/docs/resources/grant
26     alias = "field_mws"
27     host  = local.workspace_host_url["${var.environment}_field"] //digital field
28 }
29
30 provider "gitlab" {
31     alias = "databricks_group"
32     token = var.databricks_gitlab_token
33 }
34
35 provider "gitlab" {
36     alias = "digital_analytics_group"
37     token = var.digital_analytics_gitlab_token
38 }

```

and hence the remaining `catalog.tf` and `bu_grants.tf`

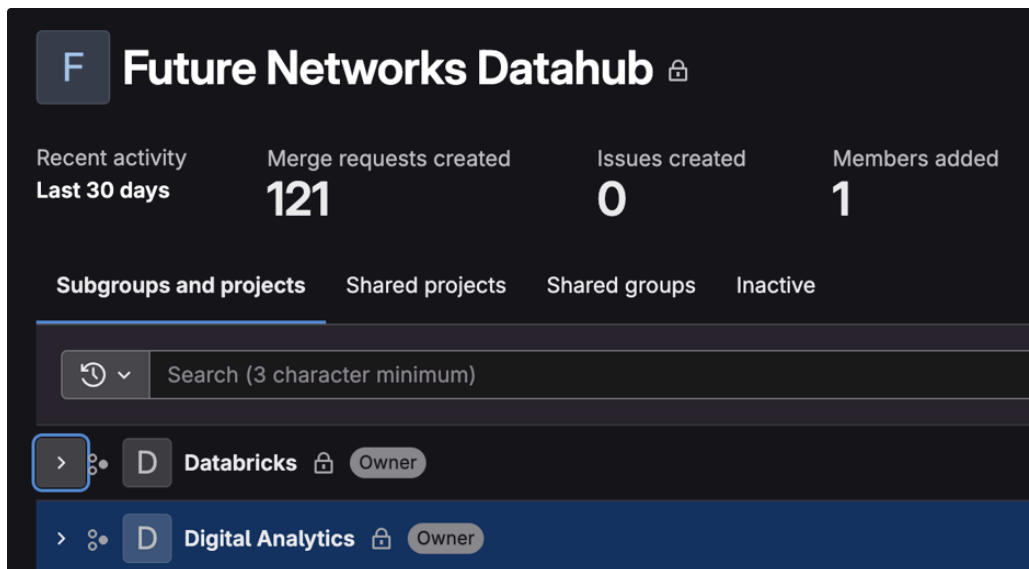
#### Gitlab Variable provisioning

To allow the following CI/CD variables to propagate across workspaces, please ensure you have the Gitlab repo group owner access.

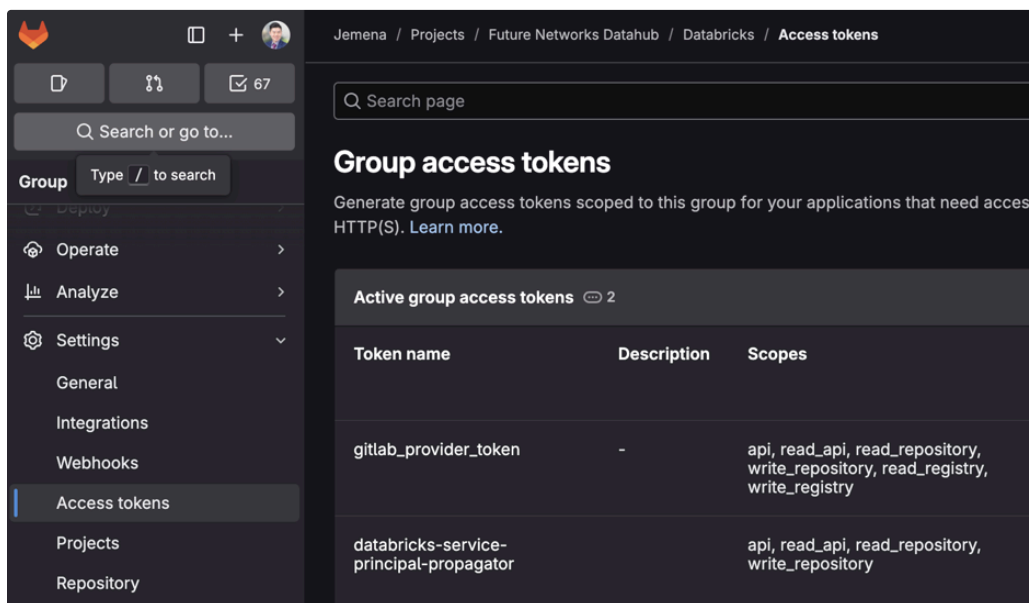
- `<BU>_<field/lab>_wks_ci_<nonprod/prod>_sp_client_id`
- `<BU>_<field/lab>_wks_ci_<nonprod/prod>_sp_client_secret`

For checking Gitlab Group Access, navigate to **Subgroup and projects >** and check if if you have `OWNER` tag on required subgroup (in this case **Databricks** and **Digital Analytics**)

if not, please contact the platform team representative to add you as a group owner



Then click through the group and navigate **Settings** (on the left-side bar tabs) > **Access Tokens** > **Add New Token** > Define **Token name** and **Select all scopes except** `ai_features`, `k8_proxy`, `manage_runner`, `write_registry` > **Create group access token**.  
**ENSURE you copy and retain the access token until the next step as pop-up window will only appear once**



Next, copy the Access Token into `databricks-unity-catalog` repo, to do so, navigate to [databricks-unity-catalog](#) > **Settings** > **Variables** > **Add variable** > Enter the the following attributes

Add Variable Attributes	Value
Visibilities	Masked and hidden

Flags	<Unselect all entries>
Descriptions	Derived from Gitlab Token < <b>Token Name</b> >
Key	<project name>_project_gitlab_token
Value	<Pasted value from previous steps>

Finally, utilise `databricks-unity-catalog` repo, navigate to `stacks/<business unit>-bu/provider.tf`, ensure the related project blocks (e.g. databricks-groups, and digital-analytics group above) are appended

```

1 provider "gitlab" {
2   alias = "databricks_group"
3   token = var.databricks_gitlab_token
4 }
5
6 provider "gitlab" {
7   alias = "digital_analytics_group"
8   token = var.digital_analytics_gitlab_token
9 }

```

and these variables are declared on `variables.tf` under related stack & CI variables declared under `.gitlab-ci.yml` for BU related pipelines below

```

1 <bussiness unit>-bu-<plan/apply>-nonprod:
2   stage: plan
3   ...
4   variables:
5     STATEFILE_NAME: "digital-bu"
6     ENV: "nonprod"
7     DATABRICKS_HOST: "https://accounts.cloud.databricks.com"
8     DATABRICKS_CLIENT_ID: $app_datahub_nonprod_databricks_account_sp_clientid
9     DATABRICKS_CLIENT_SECRET: $app_datahub_nonprod_databricks_account_sp_secret
10    DATABRICKS_ACCOUNT_ID: $app_datahub_nonprod_databricks_account_id
11    ...
12    TF_VAR_databricks_gitlab_token: $databricks_project_gitlab_token
13    TF_VAR_digital_analytics_gitlab_token: $digital_analytics_project_gitlab_token

```

## 5 Remaining Databricks Workspace-level Resources

On [Jemena / Projects / Future Networks Datahub / Databricks / databricks-workspaces](#), examining the CI/CD Group variables via

**Settings** (on the left-hand bar) > **CI/CD** > Scroll under Group variables to view the following request

- `<BU>_<field/lab>_wks_ci_<nonprod/prod>_sp_client_id`
- `<BU>_<field/lab>_wks_ci_<nonprod/prod>_sp_client_secret`
- `<BU>_<field/lab>_wks_url`

with the following screenshot as example below:

Key ↑	Value	Environments	Actions
digital_field_wks_ci_nonprod_sp_client_id digital field workspace non prod CI SP client id Masked	*****	All (default)	
digital_field_wks_ci_nonprod_sp_client_secret Masked	*****	All (default)	
digital_field_wks_url	*****	All (default)	
digital_lab_wks_ci_nonprod_sp_client_id digital lab workspace non prod CI SP client id Masked	*****	All (default)	
digital_lab_wks_ci_nonprod_sp_client_secret digital lab workspace non prod CI SP client secret Masked	*****	All (default)	
digital_lab_wks_url digital lab workspace URL	*****	All (default)	

CI/CD variables for `databricks-workspaces`

Following implementing BU stacks (example is `./stacks/digital-bu`), with `bu_<lab/field>_storage_mapper` created according to the source system aligned from the data product design for that specific BU (e.g. SAP hana would be landing in `bronze_sap_hana`):

```

1 // mapper BU medallion buckets
2 locals {
3   bu_lab_catalog_storage_mapper = {
4     "landing"      = local.bu_lab_catalog_s3_buckets["landing"].name
5     "bronze_<source system>" = local.bu_lab_catalog_s3_buckets["bronze"].name //e.g.
6     bronze_sap_hana
7     "silver"       = local.bu_lab_catalog_s3_buckets["silver"].name
8     "gold"         = local.bu_lab_catalog_s3_buckets["gold"].name
9   }
10
11   bu_field_catalog_storage_mapper = {
12     "landing"      = local.bu_field_catalog_s3_buckets["landing"].name
13     "bronze_<source system>" = local.bu_field_catalog_s3_buckets["bronze"].name
14     "silver"       = local.bu_field_catalog_s3_buckets["silver"].name
15     "gold"         = local.bu_field_catalog_s3_buckets["gold"].name
16   }
17
18   bu_qa_catalog_storage_mapper = {
19     "landing"      = local.bu_qa_catalog_s3_buckets["landing"].name
20     "bronze_<source system>" = local.bu_qa_catalog_s3_buckets["bronze"].name
21     "silver"       = local.bu_qa_catalog_s3_buckets["silver"].name
22     "gold"         = local.bu_qa_catalog_s3_buckets["gold"].name
23   }

```

Follow by configuring additional `yaml` block under `gitlab-ci.yml`, with `digital_workspaces` as example below:




```

1 nonprod digital_workspaces plan:
2   stage: plan
3   # artifacts:
4   #   paths:
5   #     - stacks/*/*.tfplan
6   variables:
7     STATEFILE_NAME: "digital-bu-wks-res"
8     ENV: "nonprod"
9     TF_VAR_business_unit: "digital"
10    TF_VAR_lab_workspace_ci_sp_id: $digital_lab_wks_ci_nonprod_sp_client_id
11    TF_VAR_lab_workspace_ci_sp_secret: $digital_lab_wks_ci_nonprod_sp_client_secret
12    TF_VAR_lab_workspace_url: $digital_lab_wks_url
13    TF_VAR_field_workspace_ci_sp_id: $digital_field_wks_ci_nonprod_sp_client_id
14    TF_VAR_field_workspace_ci_sp_secret: $digital_field_wks_ci_nonprod_sp_client_secret
15    TF_VAR_field_workspace_url: $digital_field_wks_url
16    TF_VAR_ci_job_token: $CI_JOB_TOKEN
17  before_script:
18    - *terraform-ver-init
19  script:
20    - cd stacks/digital-bu
21    - ../../scripts/nslookup.sh #debug
22    - ../../scripts/tf-plan.sh

```

## 6 Populate Allowed JARs/Init Scripts to Workspace Catalogs

To allow Maven and Init Scripts access within the Workspace catalog and compute resources, navigate to

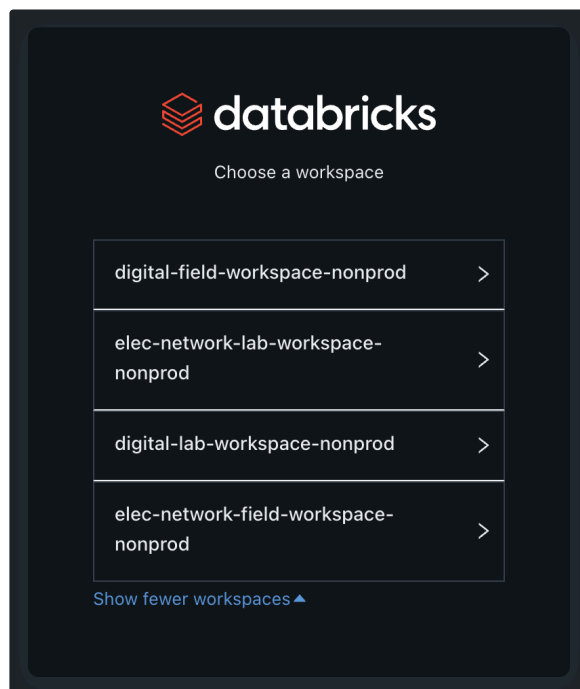
1. Click  >  >  (or any metastore with naming conventions `app-databuf-  
datahub-<prod/nonprod>-jemena-metastore`)
2. Navigate to **Allowed JARs/Init Scripts** tab, and **add** for the following:

Type	Source Type	Source
Init script	Volume	/Volumes/digital_field_catalog/global/scripts
		/Volumes/digital_lab_catalog/global/scripts

		/Volumes/digital_qa_catalog/global/scripts
		/Volumes/elec_network_field_catalog/global/scripts
		/Volumes/elec_network_lab_catalog/global/scripts
		/Volumes/elec_network_qa_catalog/global/scripts
Maven	Coordinates	com.mysql:mysql-connector-j:9.0.0
		com.sap.cloud.db.jdbc:ngdbc:2.22.11
		com.sap.cloud.db.jdbc:ngdbc:2.22.12
		org.postgresql:postgresql:42.7.4


#### Logging into Databricks workspaces

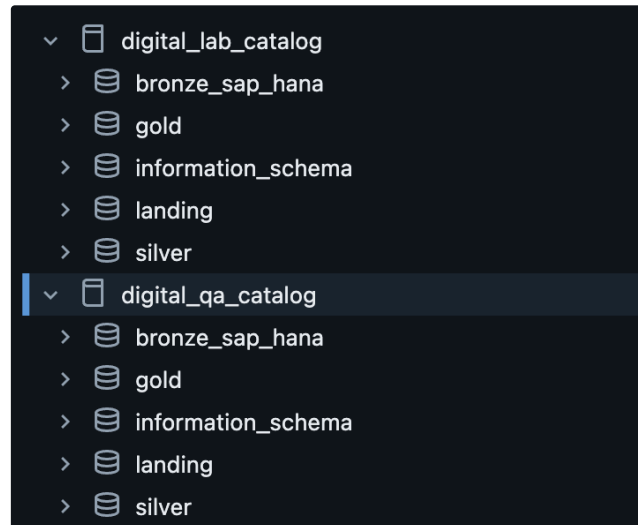
Using <https://accounts.cloud.databricks.com/> with your Jemena account (SSO sign-in), the available workspaces (subjected to RBAC) would be selectable like screenshot below:



Databricks Workspace Login Page

## Examining available Catalogs & associated backend for the given workspace

Whilst logging into workspace page, navigate to  **Catalog** icon via the column bar on your left-hand side. The available **lab** and **qa** catalogs and their managed tables for **digital-lab** workspaces (as an example can be seen as follow)



Then, click the Catalog of interests ( **bronze\_sap\_hana** under **digital\_lab\_catalog** as an example) > then click **Details** located on the top bar under managed table name, you may see the following:

Name	bronze_sap_hana
Catalog Name	digital_lab_catalog
Owner	bef2c7f1-af2a-431a-88bc-0142c04baf52
Properties	managed_by=terraform purpose=Used for bronze_sap_hana
Storage Root	s3://app-datahub-nonprod-s3-digital-lab-bronze/data
Metastore Id	9f5dba16-22ca-4d44-9cd2-b1dee875fd6a
Created At	11/10/2024, 9:56:38 pm
Created By	bef2c7f1-af2a-431a-88bc-0142c04baf52
Updated At	11/10/2024, 9:56:38 pm
Updated By	bef2c7f1-af2a-431a-88bc-0142c04baf52
Catalog Type	MANAGED_CATALOG
Storage Location	s3://app-datahub-nonprod-s3-digital-lab-bronze/data/__unitystorage/schemas/1242e3b7-5cbe-4856-b2b9-c2f10c3d1bd3
Schema Id	1242e3b7-5cbe-4856-b2b9-c2f10c3d1bd3
Browse Only	false

All tables and the associated S3 backend (i.e. **S3://...** the link next to **Storage Root**) and exhibits MANAGED TABLE type (i.e. **MANAGED\_CATALOG** type).

