Deploy Analytical Data Platform on AWS in One Day

Every business is focused on a rapid time to market and return on investment. It's no longer enough to implement a data lake, businesses require a data platform that can provide immediately actionable insights. But building a data platform from the ground up can take a significant amount of time so Grid Dynamics has developed an accelerator to help companies achieve this far more quickly. This article provides a step by step guide on how to run the accelerator on AWS from scratch.

What does Analytical Data Platform provide?

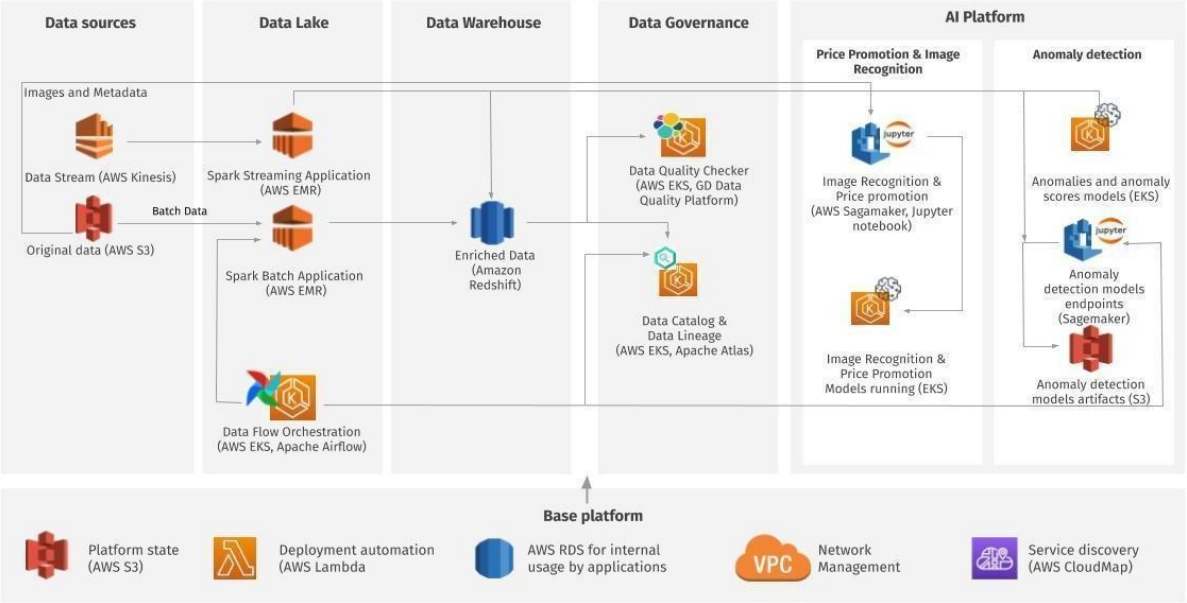
The accelerator comes as an AWS Marketplace solution or application in AWS Service Catalog.
It provides six capabilities or use cases that can be deployed on top of AWS and each capability can be provisioned separately. As they are re-using the same infrastructure, if EMR is already provisioned by one of them, it doesn't need to be provisioned again.

The potential use cases are outlined below:

- **Data lake** - is built on top of S3 with data catalog and data lineage available in Apache Atlas.
- **Enterprise data warehouse** - use case runs on top of Redshift.
- **Batch analytics** - use case covers typical batch processing and analytics using data lake, EDW, and jobs running on top of EMR and orchestrated by Apache Airflow.
- **Stream analytics** - covers the stream processing and stream analytics use cases and examples with Amazon Kinesis, Apache Spark, and pipeline orchestration with Apache Airflow.
- **Data governance** - provides tools for data catalog, data glossary, data lineage, data monitoring, and data quality. The data catalog, glossary, and lineage are implemented with Apache Atlas. Data monitoring and quality capabilities are implemented with the Kibana, ElasticSearch, Grafana, k8s, and a number of custom applications.
- **CI/CD** - the platform is deployed from scratch by CloudFormation scripts and custom lambdas.
- **Anomaly detection** - for more information about the anomaly detection architecture and technology stack, refer to a separate article on how to add anomaly detection to your data pipelines.
- **AI/ML use cases** - we included two AI use cases in the accelerator to demonstrate end-to-end functionality. All use cases are implemented with Amazon Sagemaker and Jupyter Notebooks and use the data we prepared in the data lake and EDW. One of the use cases contains a model to detect attributes in an e-commerce product catalog, while the second implements price optimization and promotion planning. We kept the use case implementations simple for the purposes of the demo.

If you're interested in production implementation of these use cases, please read articles about product attribution with image recognition and price and promotion optimization or reach out to us.

The data platform enables DataOps practices and is ready to integrate with external data sources. The high-level architecture of the platform is outlined in the diagram below:



Below we will cover all the platform's capabilities in more detail but before jumping into describing them, it's worth mentioning that the platform is distributed as either an AWS Marketplace or Service Catalog application. The difference between Marketplace and Service Catalog is that while Marketplace applications are publicly available, Service Catalog solutions are private and can be shared with other organizations.

This article will essentially be about installation from AWS Marketplace to AWS Service Catalog. As of now there is no easy way to install Marketplace container based products directly into Service Catalog (for AMI-based there is single click functionality) so as a prerequisite, detailed instruction will be provided on how to install products to Service Catalog.

Prerequisites

In AWS Marketplace, the following list of solutions are available:

1. Base platform
2. Use cases

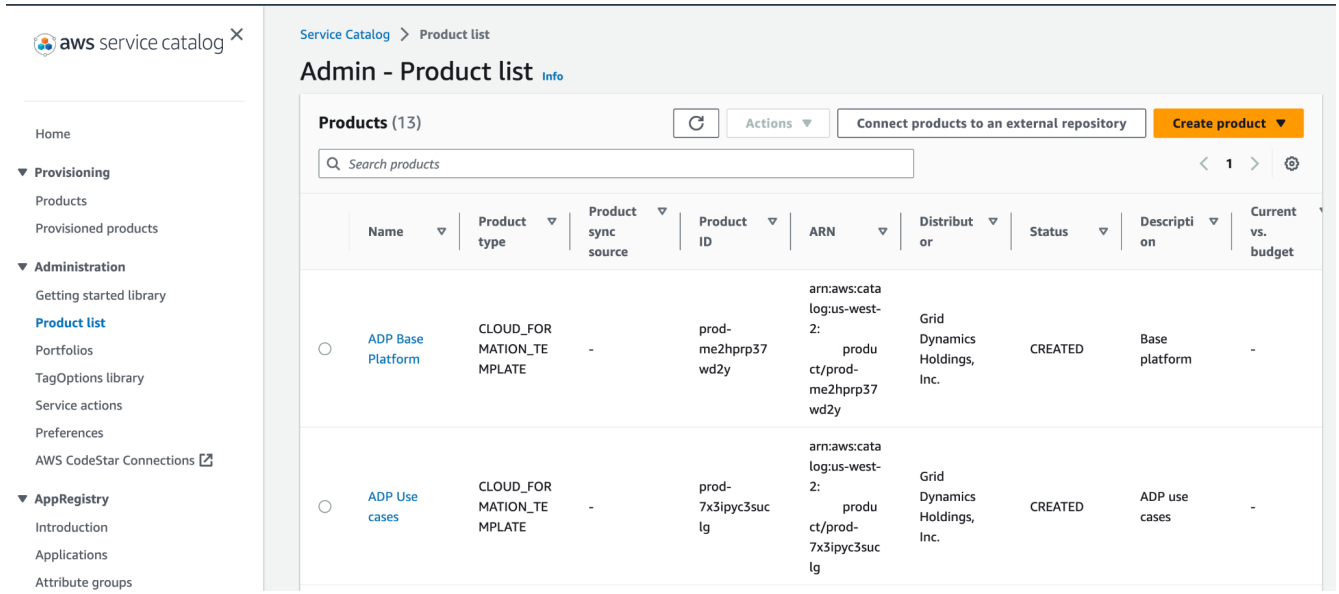
   Use cases comprises:

   1. Batch capability
   2. Streaming capability
   3. Data Quality capability
   4. Data Governance capability
   5. Dashboard capability
   6. ML capability: Visual Attributes
   7. ML capability: Promotion Planning
   8. ML capability: ML Operations

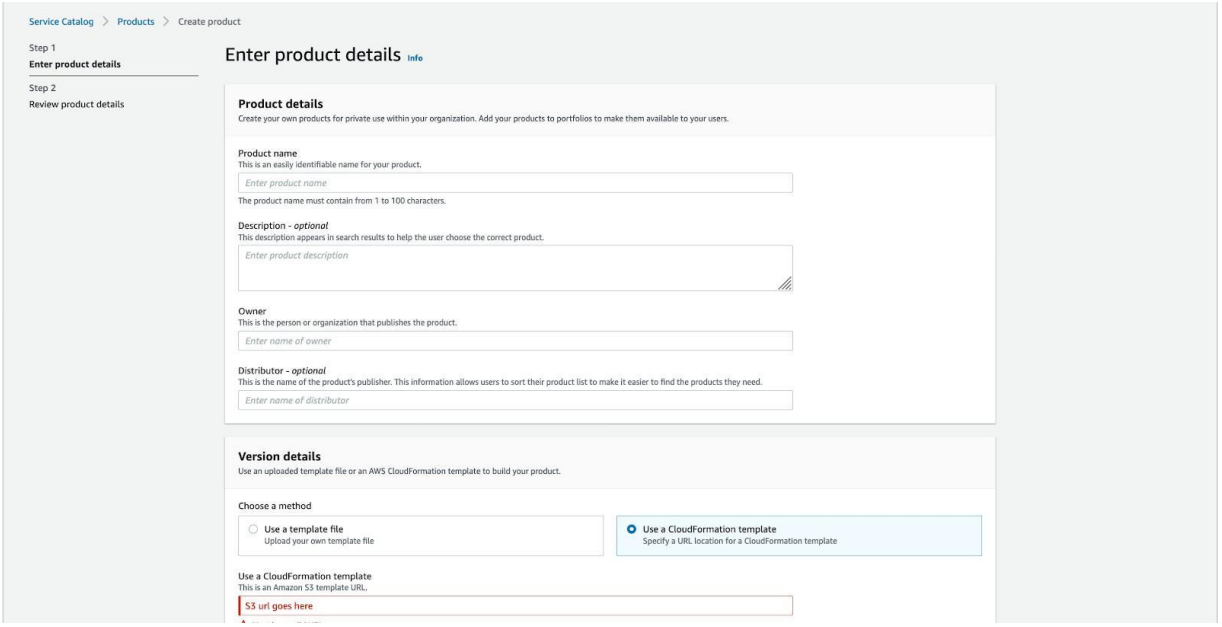
9. ML capability: ML observability

Base platform and **Use cases** on the installation page provides a link to the S3 bucket where all CloudFormation scripts reside. These products should be installed to Service Catalog in any order one by one:

1. Take the S3 URL to CloudFormation scripts
2. Go to AWS Service Catalog
3. On the left panel in the **Administration** section click on **Products list:**

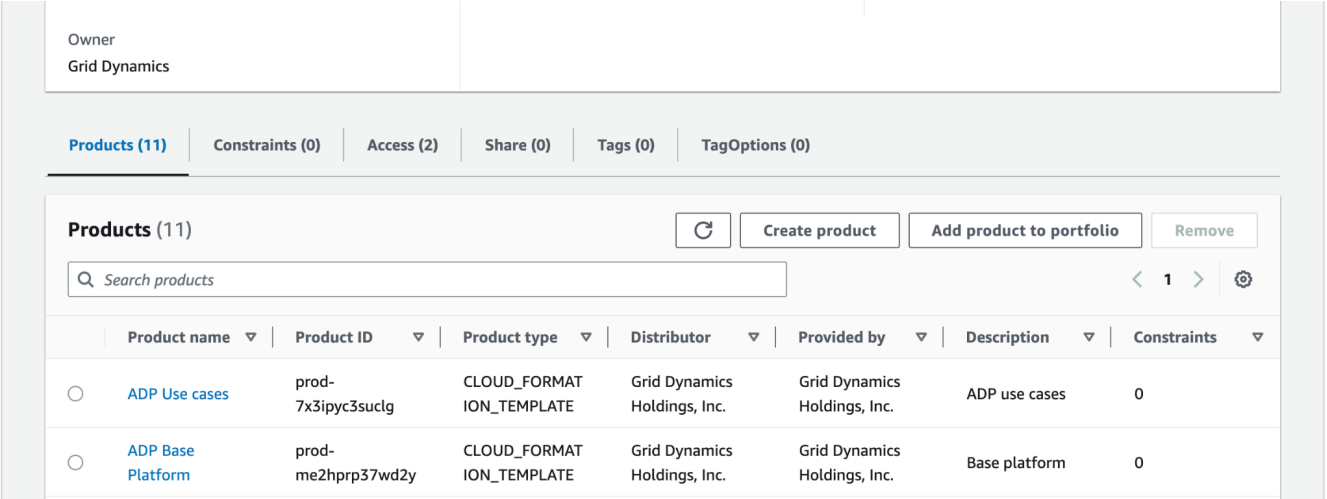


4. On the right side click to **Upload new products**:



5. Enter the product name and paste the S3 URL from Marketplace
6. You're done, the product is now created

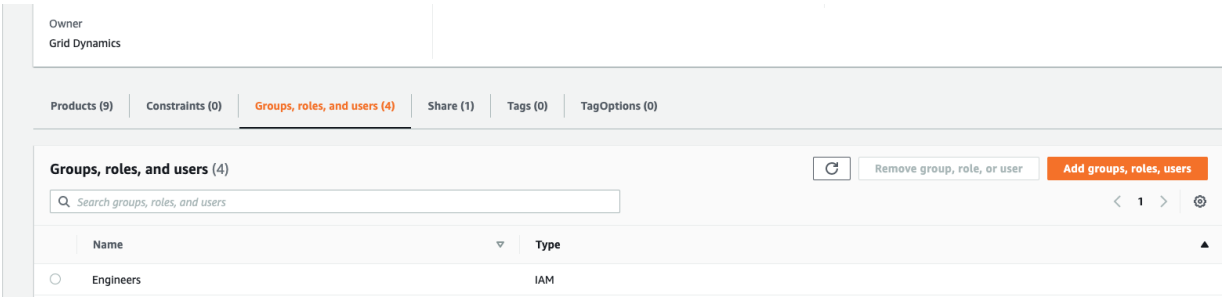
Once a product is created, it should be added to a portfolio. All permissions for a product will be managed by portfolio. To create a portfolio just click on the Portfolio link on the left panel and create a new one of re-use existing. Product listing in portfolio will look like on image below



You're done, now the final step will be to create permissions and assign to the created portfolio. There are two ways:

1. Attach a group, like engineering group
2. Or specific, not a root user

On the image below engineering group is attached:



If there are no specific groups created, follow security configuration steps below:

Security configuration

1. Create two Engineer Policies with access to ServiceCatalog and IAM actions accessPermission policy for Engineer Group might looks like below, can be adjusted if needed and replace <ACCOUNT_ID> to your account and <RESOURCE_PREFIX> to your prefix in the platform:

1.1 High level policy:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "iam:CreateRole",
            "iam:Get*",
            "iam:List*"
         ],
         "Resource": [
            "arn:aws:iam::<ACCOUNT_ID>:role/*"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AddUserToGroup",
                "iam:CreateGroup",
                "iam:DeleteGroup",
                "iam:DeleteGroupPolicy",
                "iam:GetGroup",
                "iam:GetGroupPolicy",
                "iam:PutGroupPolicy"
            ],
            "Resource": [
                "arn:aws:iam::<ACCOUNT_ID>:group/*ADPMLEngineer*",
                "arn:aws:iam::<ACCOUNT_ID>:group/*ADPDQEngineer*",
                "arn:aws:iam::<ACCOUNT_ID>:group/*ADPDevOpsEngineer*",
                "arn:aws:iam::<ACCOUNT_ID>:group/*ADPBigDataEngineer*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy”,
                "iam:DeleteRolePolicy",
                "iam:DeleteRole"
            ],
            "Resource": [
                "arn:aws:iam::<ACCOUNT_ID>:role/SC-<ACCOUNT_ID>-pp-*",
                "arn:aws:iam::<ACCOUNT_ID>:role/*-Churn-Predictions-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
                "iam:PutRolePolicy"
            ],
            "Resource": [
                "arn:aws:iam::<ACCOUNT_ID>:role/SC-<ACCOUNT_ID>-pp-*",
                "arn:aws:iam::<ACCOUNT_ID>:role/eks-quickstart-*",
                "arn:aws:iam::<ACCOUNT_ID>:role/<RESOURCE_PREFIX>-*",
                "arn:aws:iam::<ACCOUNT_ID>:role/<RESOURCE_PREFIX>-Churn-Predictions-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action":
            "s3:GetObject",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/servicecatalog:provisioning": "true"
                }
            }
        }
    ]
}
```

1.2 Medium level policy:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "elasticmapreduce:RunJobFlow",
            "kms:Decrypt",
            "kms:GenerateDataKey",
            "lambda:CreateFunction",
            "lambda:DeleteFunction",
            "lambda:GetFunction",
            "lambda:InvokeFunction",
            "lambda:UpdateFunctionCode",
            "rds:AddTagsToResource",
            "rds:CreateDBInstance",
            "rds:CreateDBSubnetGroup",
            "rds:DeleteDBInstance",
            "rds:DescribeDBInstances",
            "rds:DeleteDBSubnetGroup",
            "rds:DescribeDBSubnetGroups",
            "rds:ListTagsForResource",
            "rds:RemoveTagsFromResource",
            "sagemaker:CreateModel",
            "sagemaker:CreateNotebookInstance",
            "sagemaker:DeleteEndpoint",
            "sagemaker:DeleteEndpointConfig",
            "sagemaker:DeleteModel",
            "sagemaker:DeleteNotebookInstance",
            "sagemaker:DescribeEndpoint",
            "sagemaker:DescribeModel",
            "sagemaker:DescribeNotebookInstance",
            "sagemaker:StartNotebookInstance",
            "sagemaker:UpdateNotebookInstance",
            "sagemaker:*NotebookInstanceLifecycleConfig",
            "secretsmanager:GetSecretValue",
            "secretsmanager:TagResource",
            "secretsmanager:UntagResource",
            "servicediscovery:CreateService",
            "servicediscovery:DeleteService",
            "s3:Get*",
            "s3:ListBucket",
            "s3:ListBucketMultipartUploads",
            "s3:ListBucketVersions",
            "s3:ListMultipartUploadParts",
            "s3:PutEncryptionConfiguration"
         ],
         "Resource": [
            "arn:aws:kms:*:<ACCOUNT_ID>:key/*",
            "arn:aws:lambda:*:<ACCOUNT_ID>:function:eks-quickstart-*",
            "arn:aws:lambda:*:<ACCOUNT_ID>:function:SC-*",
            "arn:aws:lambda:*:<ACCOUNT_ID>:function:QuickStart*",
            "arn:aws:lambda:*:<ACCOUNT_ID>:function:EKS-QuickStart-*",
            "arn:aws:rds:*:<ACCOUNT_ID>:db:*",
            "arn:aws:rds:*:<ACCOUNT_ID>:subgrp:*-ai4ops-rds",
            "arn:aws:rds:*:<ACCOUNT_ID>:subgrp:*-dqweb-rds",
            "arn:aws:servicediscovery:*:<ACCOUNT_ID>:service/*",
            "arn:aws:servicediscovery:*:<ACCOUNT_ID>:namespace/*",
```

```
            "arn:aws:sagemaker:*:<ACCOUNT_ID>:notebook-instance-lifecycle-config/*",
            "arn:aws:sagemaker:*:<ACCOUNT_ID>:endpoint/*",
            "arn:aws:sagemaker:*:<ACCOUNT_ID>:model/*",
            "arn:aws:sagemaker:*:<ACCOUNT_ID>:endpoint-config/*",
            "arn:aws:sagemaker:*:<ACCOUNT_ID>:notebook-instance/*",
            "arn:aws:secretsmanager:*:<ACCOUNT_ID>:secret:*",
            "arn:aws:s3:::griddynamics-analytical-data-platform-snapshots",
            "arn:aws:s3:::griddynamics-analytical-data-platform-releases",
            "arn:aws:s3:::griddynamics-analytical-data-platform-snapshots/*",
            "arn:aws:s3:::griddynamics-analytical-data-platform-releases/*",
            "arn:aws:s3:::cf-templates-*",
            "arn:aws:s3:::sc-*-pp-*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "kinesis:AddTagsToStream",
            "kinesis:CreateStream",
            "kinesis:DeleteStream",
            "kinesis:DescribeStream*",
            "kinesis:ListTagsForStream",
            "kinesis:RemoveTagsFromStream",
            "lambda:GetLayerVersion",
            "lambda:PublishLayerVersion",
            "secretsmanager:CreateSecret",
            "secretsmanager:DeleteSecret",
            "s3:CreateBucket",
            "s3:DeleteBucket",
            "s3:ListBucket",
            "s3:PutBucketTagging"
        ],
        "Resource": [
            "arn:aws:kinesis:*:<ACCOUNT_ID>:stream/*-ai4ops-stream",
            "arn:aws:lambda:*:<ACCOUNT_ID>:layer:*",
            "arn:aws:secretsmanager:*:<ACCOUNT_ID>:secret:*",
            "arn:aws:s3:::*-batch-*",
            "arn:aws:s3:::*-streaming-*",
            "arn:aws:s3:::*-pipeline-*",
            "arn:aws:s3:::*-platform-state",
            "arn:aws:s3:::sc-*-pp-*",
            "arn:aws:s3:::cf-templates-*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": "lambda:DeleteLayerVersion",
        "Resource": "arn:aws:lambda:*:<ACCOUNT_ID>:layer:*:*"
    },
    {
        "Effect": "Allow",
        "Action": "s3:PutObject",
        "Resource": "arn:aws:s3:::cf-templates-*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "cloudformation:CreateChangeSet",
            "cloudformation:ListStacks",
            "ec2:AllocateAddress",
            "ec2:AssociateDhcpOptions",
```

"ec2:AssociateRouteTable",
"ec2:AttachInternetGateway",
"ec2:AttachNetworkInterface",
"ec2:AuthorizeSecurityGroupEgress",
"ec2:AuthorizeSecurityGroupIngress",
"ec2:CreateDhcpOptions",
"ec2:CreateInternetGateway",
"ec2:CreateKeyPair",
"ec2:CreateNatGateway",
"ec2:CreateNetworkInterface",
"ec2:CreateNetworkInterfacePermission",
"ec2:CreateRoute",
"ec2:CreateRouteTable",
"ec2:CreateSecurityGroup",
"ec2:CreateSubnet",
"ec2:CreateTags",
"ec2:CreateVpc",
"ec2:CreateVpcEndpoint",
"ec2:DeleteDhcpOptions",
"ec2:DeleteInternetGateway",
"ec2:DeleteNatGateway",
"ec2:DeleteNetworkAcl",
"ec2:DeleteNetworkAclEntry",
"ec2:DeleteNetworkInterface",
"ec2:DeleteRoute",
"ec2:DeleteRouteTable",
"ec2:DeleteSecurityGroup",
"ec2:DeleteSubnet",
"ec2:DeleteVpc",
"ec2:DeleteVpcEndpoints",
"ec2:Describe*",
"ec2:DetachInternetGateway",
"ec2:DetachNetworkInterface",
"ec2:DisassociateRouteTable",
"ec2:ModifySubnetAttribute",
"ec2:ModifyVpcAttribute",
"ec2:ReleaseAddress",
"ec2:RevokeSecurityGroupEgress",
"ec2:RevokeSecurityGroupIngress",
"ec2:UpdateSecurityGroupRuleDescriptionsEgress",
"ec2:UpdateSecurityGroupRuleDescriptionsIngress",
"health:DescribeEventAggregates",
"lambda:ListFunctions",
"route53:ChangeResourceRecordSets",
"route53:CreateHealthCheck",
"route53:CreateHostedZone",
"route53:DeleteHealthCheck",
"route53:GetHealthCheck",
"route53:GetHostedZone",
"route53:ListHostedZonesByName",
"route53:UpdateHealthCheck",
"servicediscovery:DeleteNamespace",
"servicediscovery:DeregisterInstance",
"servicediscovery:DiscoverInstances",
"servicediscovery:CreateHttpNamespace",
"servicediscovery:CreatePrivateDnsNamespace",
"servicediscovery:CreatePublicDnsNamespace",
"servicediscovery:Get*",
"servicediscovery:List*",

```
            "servicediscovery:RegisterInstance",
            "servicediscovery:TagResource",
            "servicediscovery:UntagResource",
            "secretsmanager:GetRandomPassword",
            "s3:GetAccessPoint",
            "s3:GetAccountPublicAccessBlock",
            "s3:ListAccessPoints",
            "s3:ListAccessPointsForObjectLambda",
            "s3:ListAllMyBuckets",
            "s3:ListJobs",
            "s3:ListMultiRegionAccessPoints",
            "s3:ListStorageLensConfigurations"
        ],
        "Resource": "*"
    }
  ]
}
```

1.3 Low level policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codepipeline:CreatePipeline",
        "codepipeline:DeletePipeline",
        "codepipeline:GetPipeline",
        "codepipeline:GetPipelineState",
        "codepipeline:TagResource",
        "codepipeline:UntagResource",
        "codepipeline:UpdatePipeline"
      ],
      "Resource": [
        "arn:aws:codepipeline:*:<ACCOUNT_ID>:*-mlflow-BuildPipeline",
        "arn:aws:codepipeline:*:<ACCOUNT_ID>:actiontype:*/*/*/*",
        "arn:aws:codepipeline:*:<ACCOUNT_ID>:webhook:*",
        "arn:aws:codepipeline:*:<ACCOUNT_ID>:*-mlflow-BuildPipeline/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:ImportSourceCredentials",
        "codebuild:DeleteOAuthToken",
        "codebuild:DeleteSourceCredentials",
        "codebuild:ListBuilds",
        "codebuild:ListBuildBatches",
        "codebuild:ListConnectedOAuthAccounts",
        "codebuild:ListCuratedEnvironmentImages",
        "codebuild:ListProjects",
        "codebuild:ListReportGroups",
        "codebuild:ListReports",
        "codebuild:ListRepositories",
```

```
            "codebuild:ListSharedProjects",
            "codebuild:ListSharedReportGroups",
            "codebuild:ListSourceCredentials",
            "codebuild:PersistOAuthToken",
            "codepipeline:ListPipelines",
            "codestar-connections:CreateConnection",
            "codestar-connections:DeleteConnection",
            "codestar-connections:GetConnection",
            "codestar-connections:ListTagsForResource",
            "codestar-connections:PassConnection",
            "codestar-connections:TagResource",
            "elasticmapreduce:DescribeCluster",
            "elasticmapreduce:ListInstanceGroups",
            "elasticmapreduce:RunJobFlow",
            "elasticmapreduce:TerminateJobFlows"
         ],
         "Resource": "*"
      },
      {
         "Effect": "Allow",
         "Action":
         "codebuild:*",
         "Resource": [
            "arn:aws:codebuild:*:<ACCOUNT_ID>:project/<RESOURCE_PREFIX>-*",
            "arn:aws:codebuild:*:<ACCOUNT_ID>:report-group/*"
         ]
      }
   ]
}
```

***Important*!** Don't forget to change <ACCOUNT_ID> to your account and <RESOURCE_PREFIX> to your prefix on the platform!

2. Create an **Engineer** group, attach new created policies and attach *AWSServiceCatalogAdminFullAccess* policy to the group.

3. Create a new user (engineer) for ServiceCatalog and CFN usage (do not use root user!).

4. Add a new user to **Engineer** group.

5. Create a key pair for capability for SSH access to the bastion host.

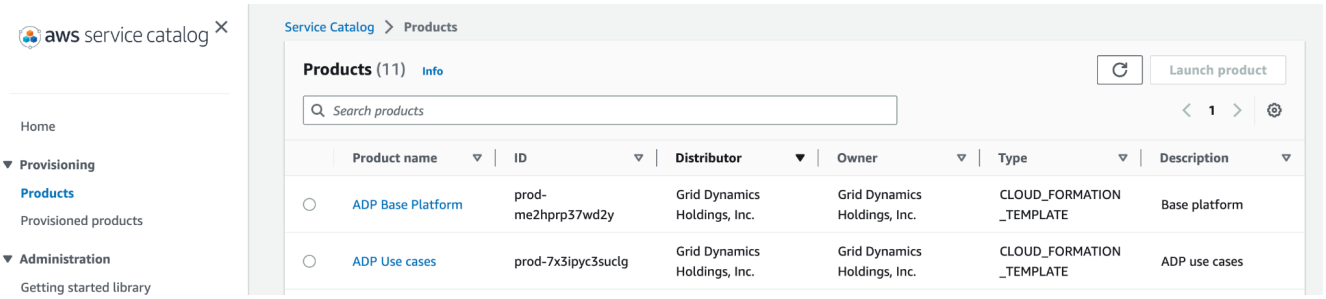
6. Login in AWS under created user.

Warning:

> *The deployment template will create IAM roles that has the ability to create additional IAM roles that may or may not include administrator permissions to the customer account where it is deployed.*

All further actions will be done in Service Catalog in AWS console

All security configurations are done, you can proceed to run the platform with service catalog applications looking like on the image:



How to access platform web services?

There are several ways how to access any web service which is running in private AWS network:

1. Share the same VPN network with common DNS
2. AWS Private link
3. Access AWS services through SSH tunneling - AWS manual

Platform has capability to expose all private web applications to a public network, but it was designed and used only for demo purpose, we highly recommend to keep all services in a private network.

Base platform

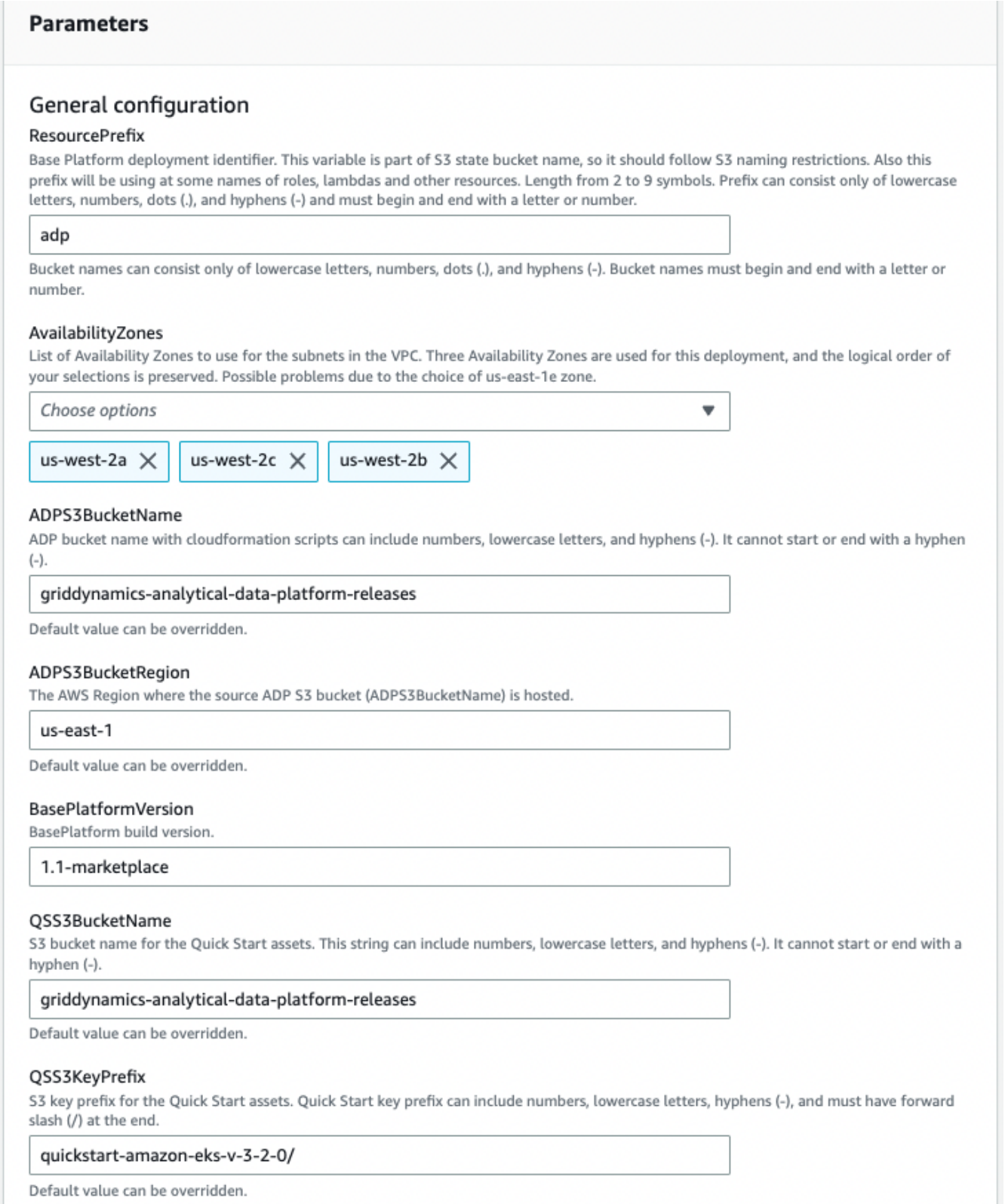
The Analytical Data Platform has a base platform responsible for provisioning networks, S3 buckets with CloudFormation code and Lambdas, and a common state, which is used to share common services like EMR or EKS between use cases. Once the base platform is in place any capability can be deployed in any order.

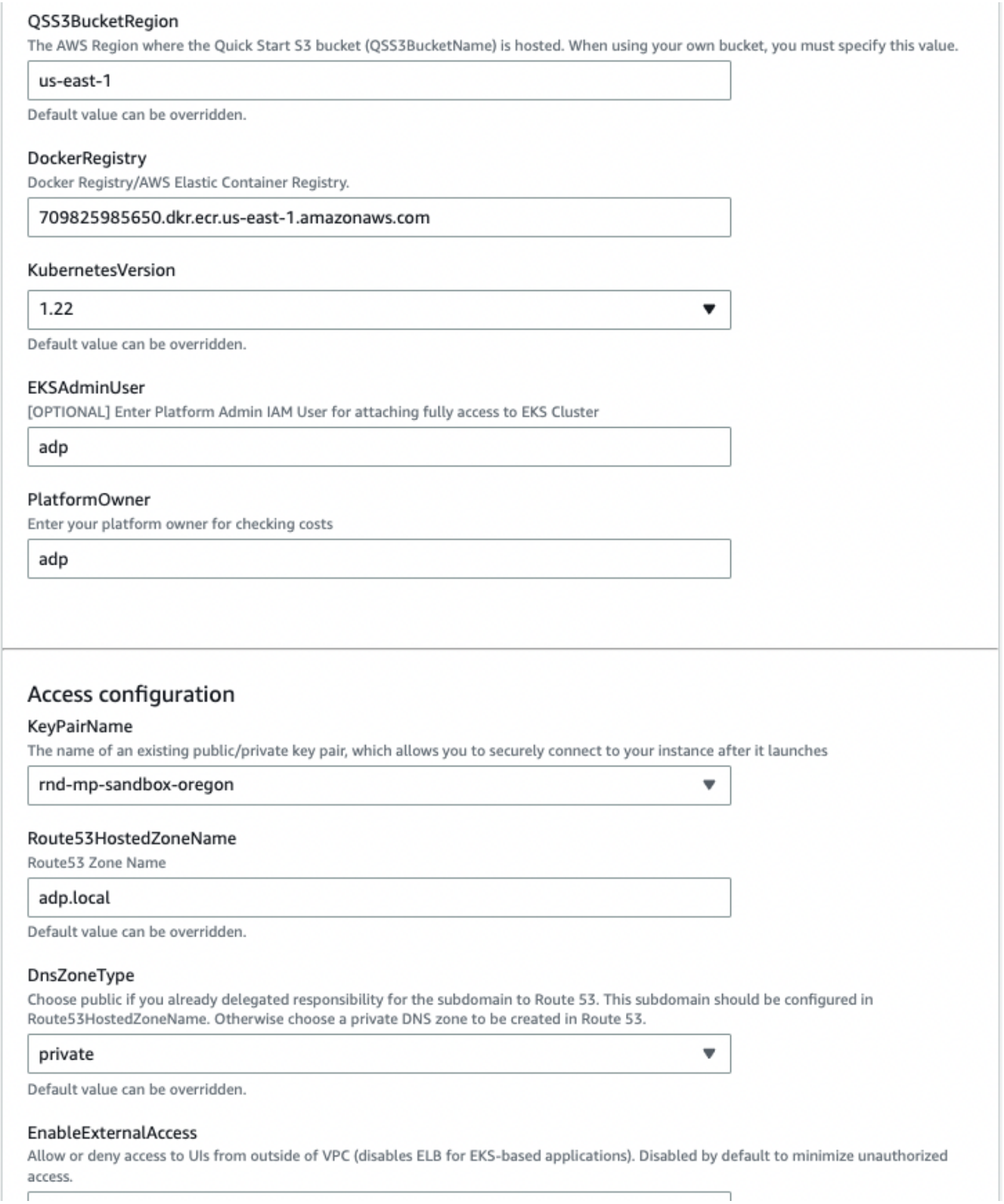
To launch the base platform, four parameters should be defined:

- Resource prefix
- SSH keys to use
- Remote access CIDR
- Availability zones - at least three should be chosen

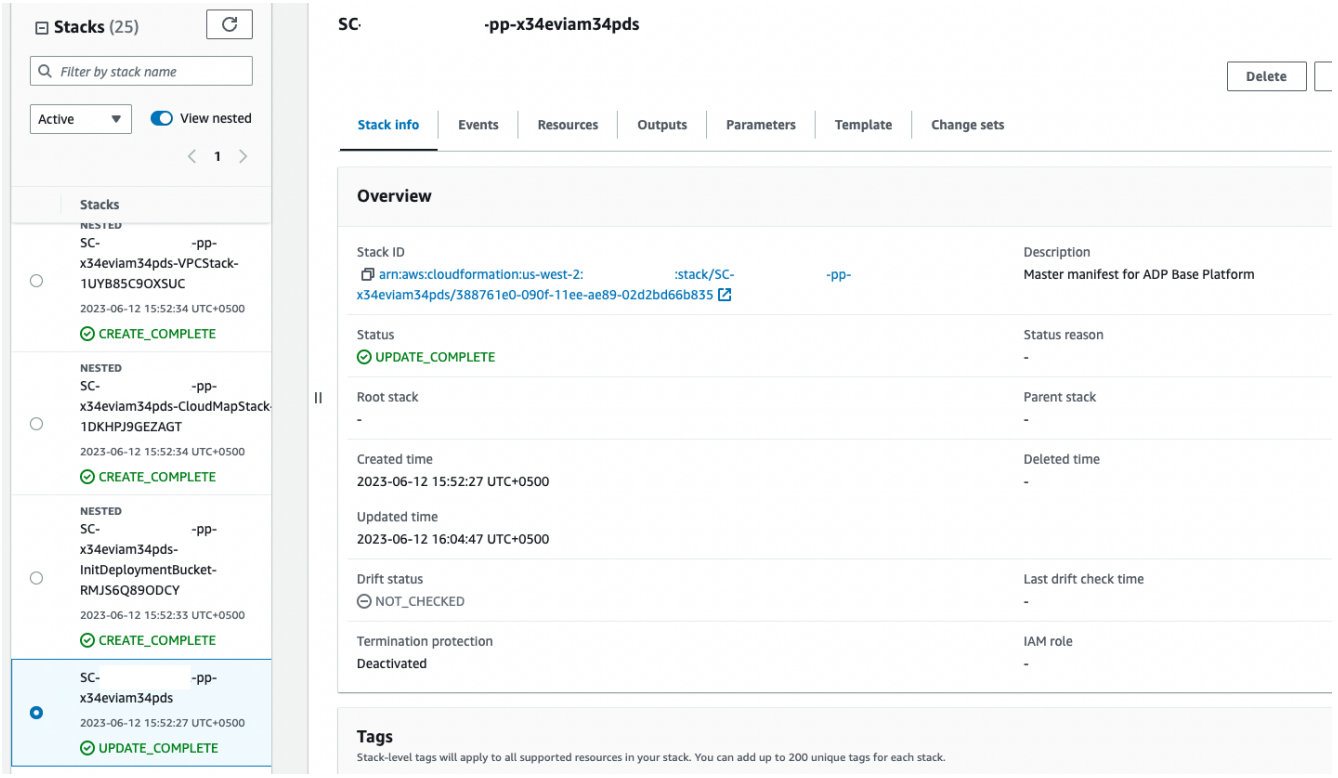
EKS requires at least three availability zones to be present in the AWS region, so before deploying the data platform please ensure that constraint is met. Further information regarding availability zones can be found on the AWS status page.

In the example below, the resource prefix is *adp-base-platform*, ssh key pair is *test*, remote access CIDR is allowing all IP addresses, and there are three availability zones defined:





The four parameters above are enough to run the platform from scratch in any AWS account. All other parameters used in the platform can be left as is. Once the base platform starts provisioning in CloudFormation, all steps will be in progress and on completion it will look like:

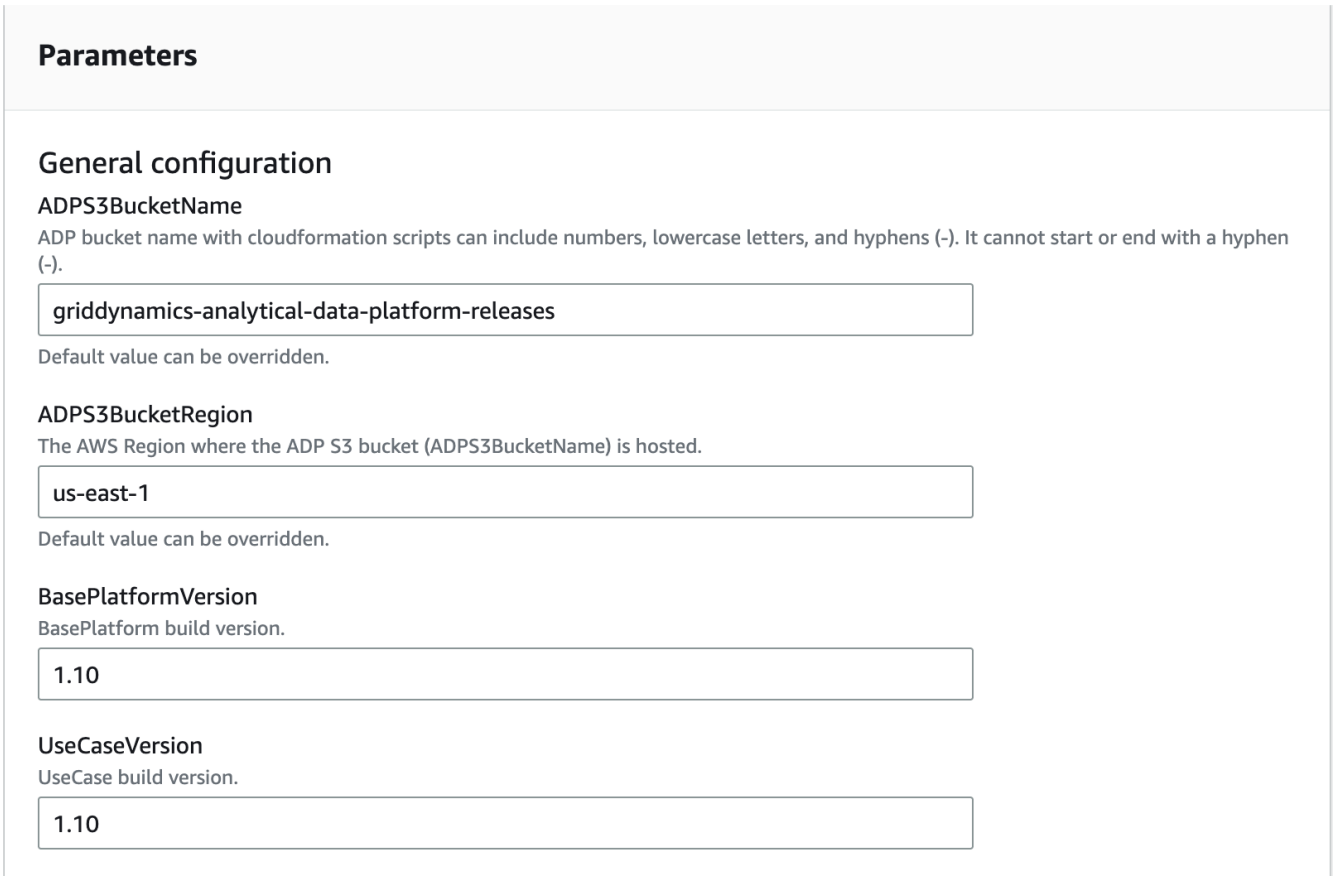


Once the base platform is up and running, any capability can be deployed on top.

Notice:
Many of our clients have base platform failing at the very beginning due to the VPC limit in AWS. Please ensure there is at least one VPC network could be created before running the base platform.

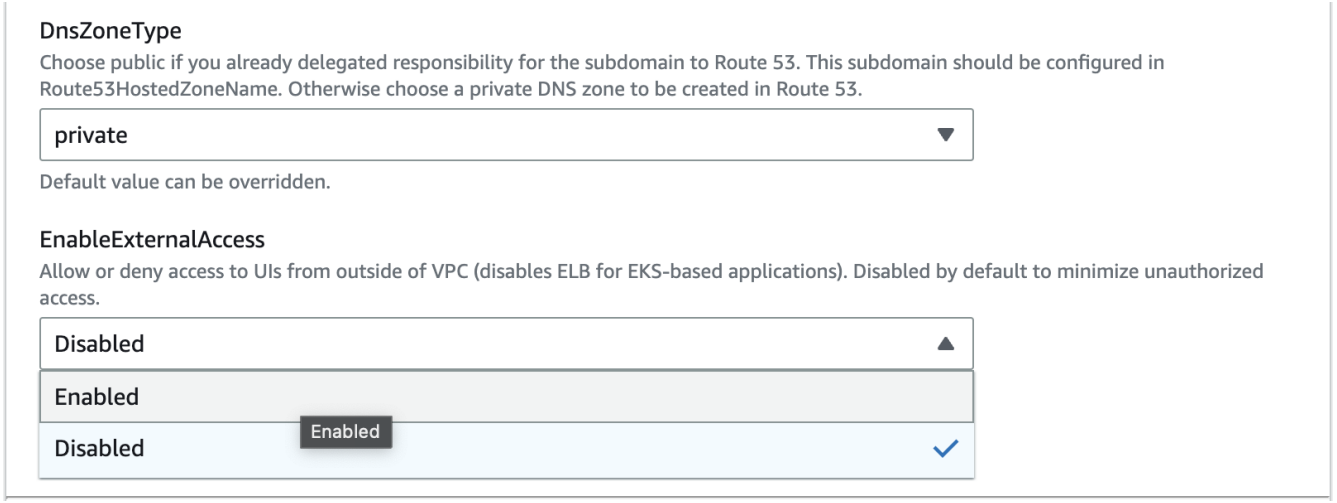
Use cases

Use cases provisions the following abilities: Batch capability, Streaming capability, Data Quality capability, Data Governance capability, Dashboard capability, ML capability: Visual Attributes, ML capability: Promotion Planning, ML capability: ML Operations. Use cases can be provisioned on top of the base platform without needing to define any parameters:



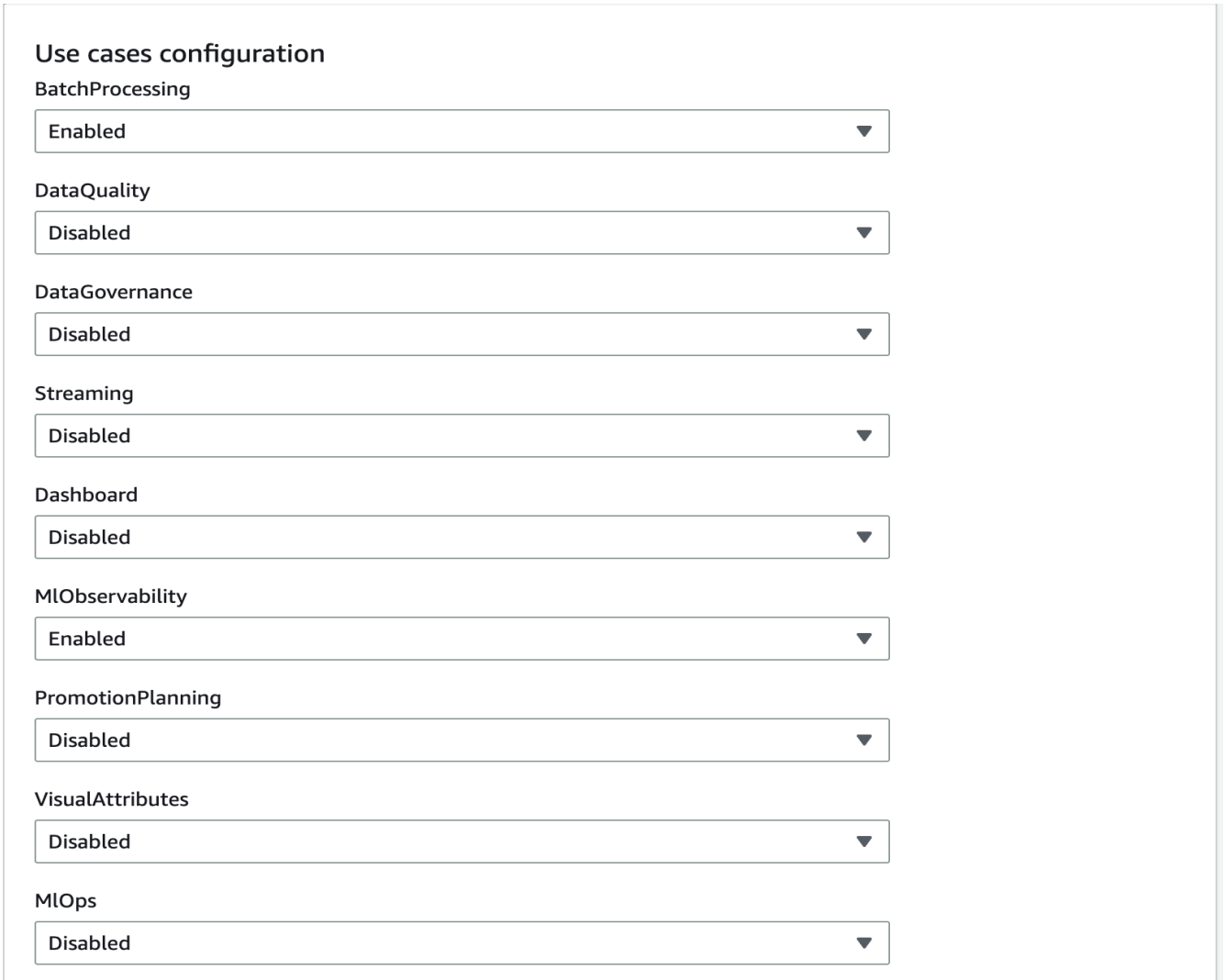
In the use cases there are two URLs: one is active inside the VPN network, the other ELB (access can be granted for the world wide internet) is active only if it's enabled. Also this item should be included in the base platform.

EnableExternalAccess is where it can be enabled.



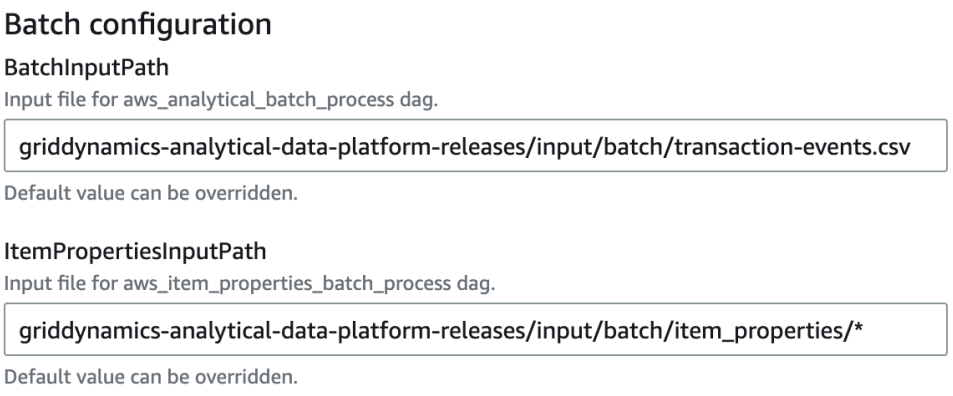
We don't recommend exposing to the public network the services, it could be used for a demo purpose for a limited time period.

In "Use cases configuration" you choose which capabilities to include:



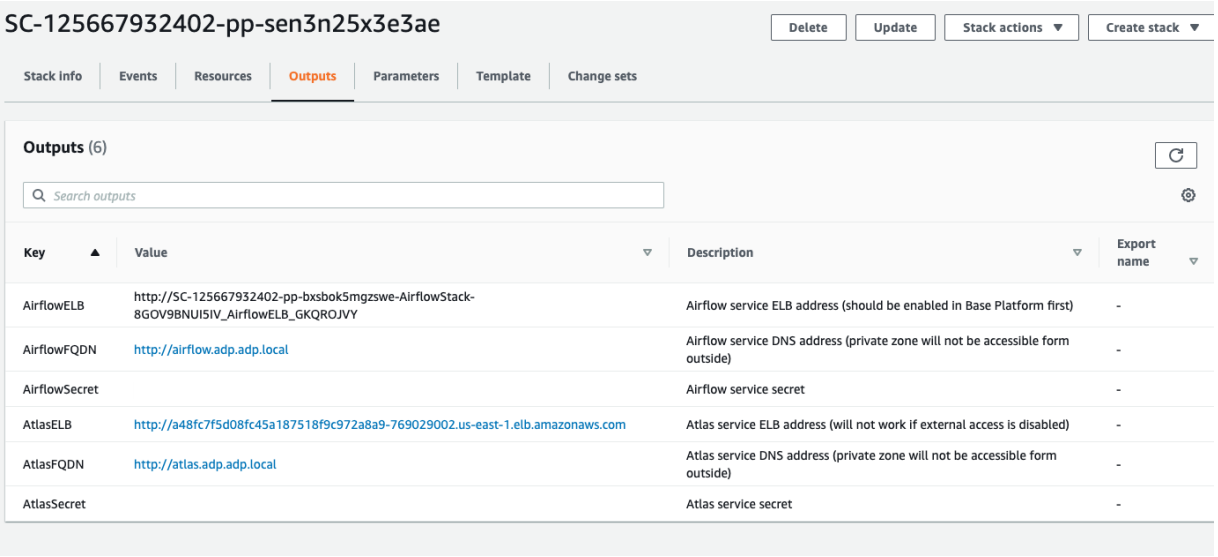
Batch analytics

Batch analytics provisions the following components: EMR for batch analytics, Apache Airflow to orchestrate the jobs, and data lake on top of S3. Batch capability can be provisioned on top of the base platform without needing to define any parameters:



BatchInputPath and ItemPropertiesInputPath are used in demo code that is running on top of the platform. Once the batch use case is up and running, demo Spark jobs can be run from the Apache Airflow UI. A link to the Apache Airflow UI will be available in CloudFormation in the deployed stack

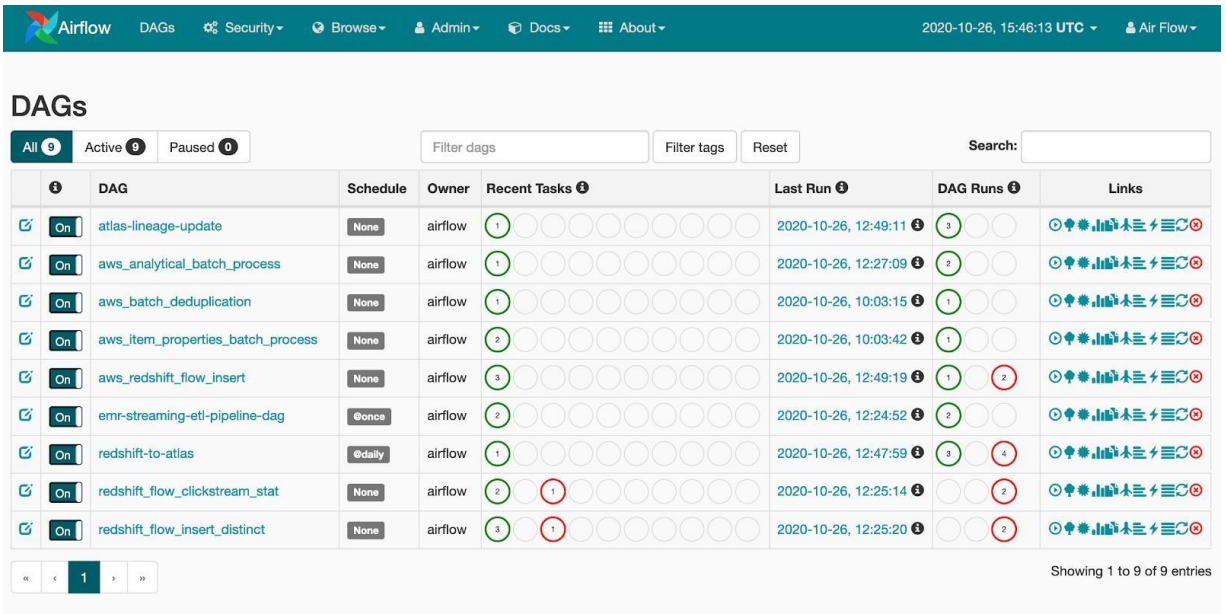
details:



In the image above there are two Airflow URLs: one is active inside the VPN network, the other (AirflowELB) is active only if it's enabled in batch or streaming capability. EnableExternalAccess is where it can be enabled.

We don't recommend exposing to the public network the services, it could be used for a demo purpose for a limited time period.

Once the batch capability is up and running you can open Apache Airflow UI or EMR and check flows running. There are several batch and streaming jobs which are deployed along with capability:



Streaming analytics

The streaming analytics capability targets real time scenarios including real time analytics

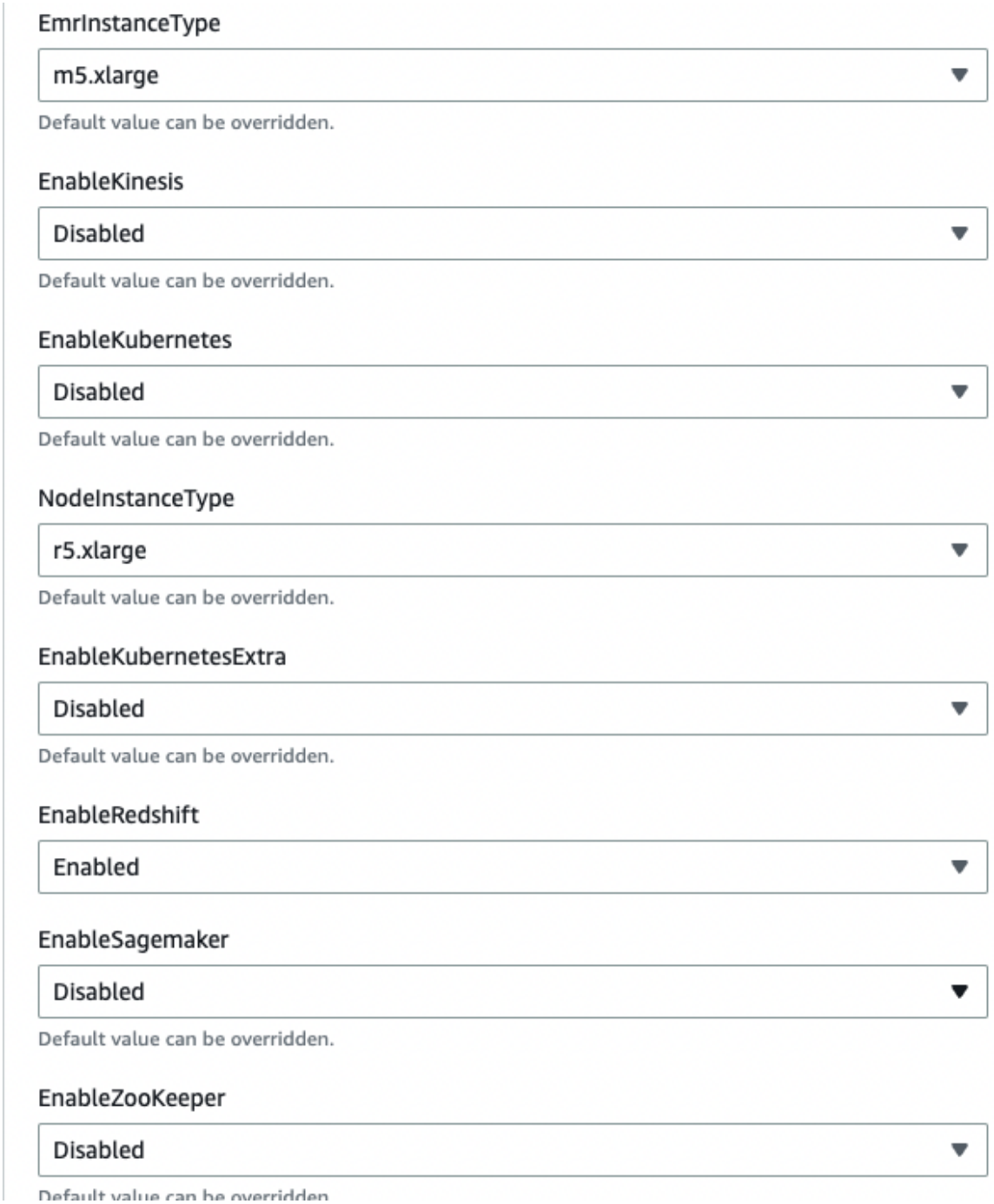
and streaming fraud detection. For the streaming analytics capability EMR, Kinesis, Apache Airflow, and S3 are provisioned. There are also streaming jobs that are orchestrated by Apache Airflow and can be used for example or demo purposes. The use case can be provisioned with zero parameters specified

---

InputFilePath is used for demo applications and is available with the use case. The rest of the configurations are system ones and should be left unchanged. In addition to this, infrastructure applications are also deployed. There are several Spark batch and streaming applications along with Airflow DAGs deployed for orchestration purposes.

Enterprise data warehouse

The analytical data platform leverages Redshift as an enterprise data warehouse solution. The platform doesn't provide a dedicated capability to deploy Redshift as an independent capability - it comes with batch, streaming, or ML use cases. Typically data platforms start with data lake and batch or streaming processing before moving onto EDW. In cases where EDW needs to be deployed without batch, streaming, or ML use cases, it can be deployed directly by enabling it in the base platform:



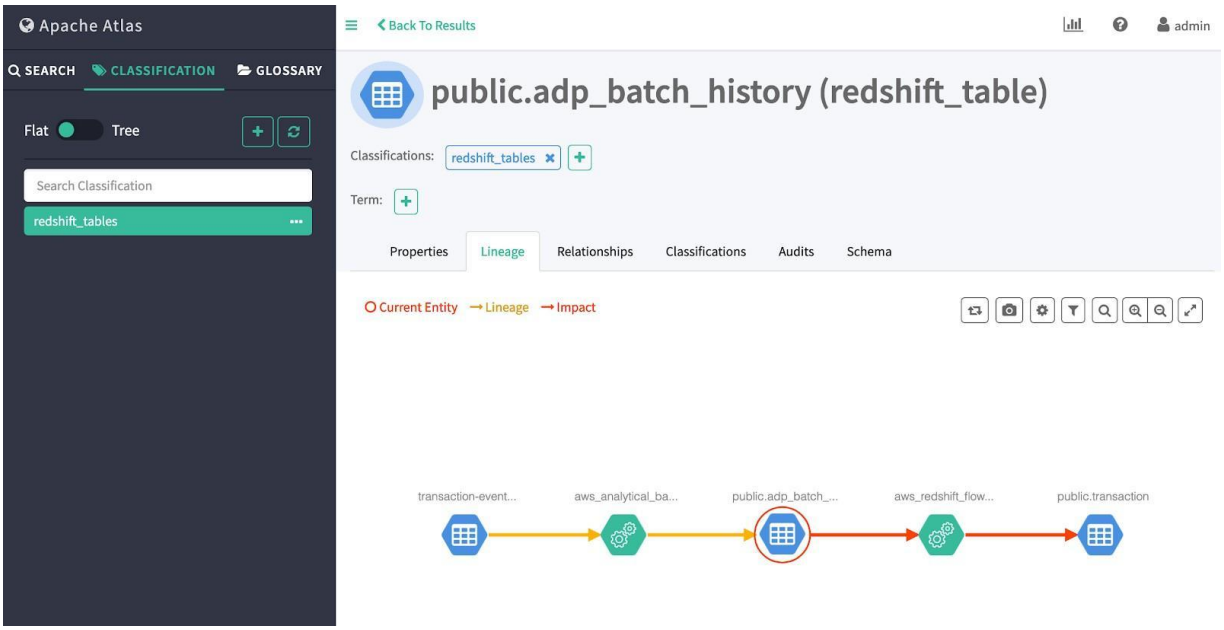
All other services are disabled and will be deployed by use cases directly.

Data governance

One of the key features of the analytical data platform is data governance integration. The platform provides tooling to setup data governance in a process similar to CI/CD:

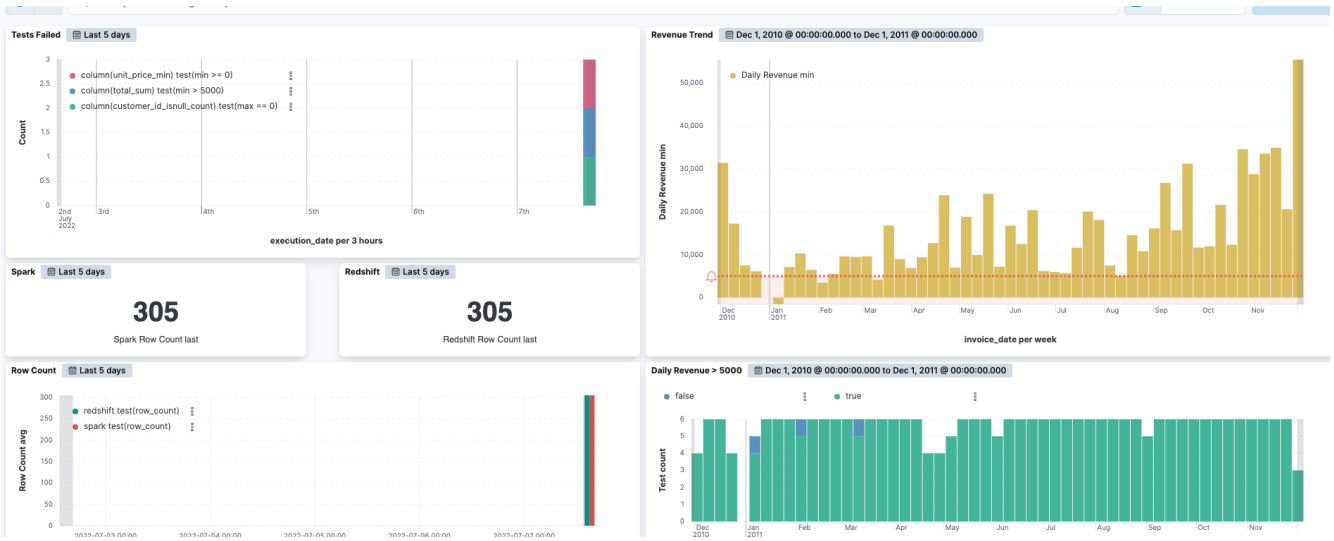
- Apache Airflow is used for workflow orchestration
- Apache Atlas is used for data catalog and data lineage
- Data quality is a custom Grid Dynamics solution

Apache Atlas builds the data catalog over all data in S3 and Redshift. Integration of Atlas and Airflow brings lineage information about all flows - Atlas provides information about what source datasets are used in the final dataset. Integration is done as a custom plugin for Airflow, which sends lineage information to Atlas where it's visualized:



Data quality is provided by Grid Dynamics' custom solution, which helps to:

1. Run data monitoring checks
2. Identify anomalies in the data
3. Run complex data quality rules
4. Check data in various data sources like S3, Redshift, Snowflake, Teradata, and others
5. Build dashboards on top of data checks
6. Send alerts Kibana dashboard:



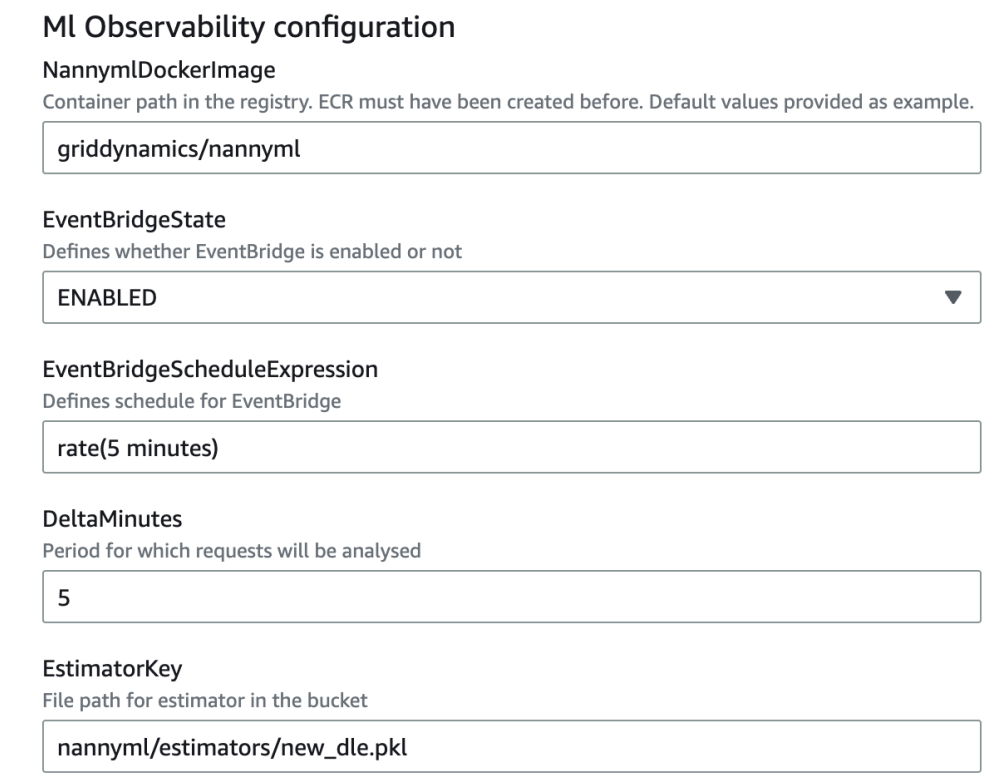
Data governance provisioning is similar to the use cases detailed above.

There are two user input parameters responsible for external access, which obviously should be private by default. The rest of the configuration is maintained by CloudFormation. Once the platform is up and running, it provides access to Apache Atlas and data quality only for VPN restricted networks. To enable world wide access please enable EnableExternalAccess in use cases configuration and base platform.

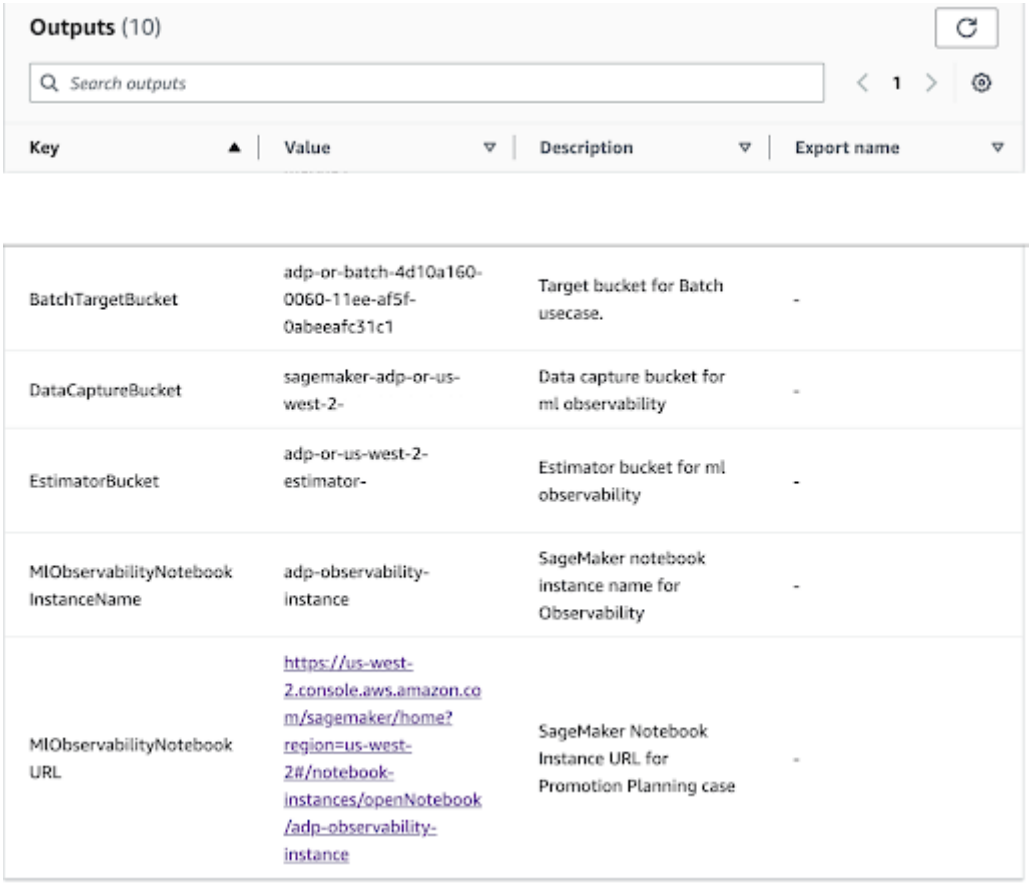
ML observability

ML Observability is an extension to ML platform aimed to help with definition of ML models' monitoring scenarios, drifts in datasets and ML models' predictions.

ML Observability provisions the following components: Sagemaker Notebook and S3 buckets. ML Observability capability can be provisioned on top of the base platform without needing to define any parameters:



In the outputs of the stack there will be a link to the created Sagemaker notebook and s3 buckets.



Outputs (10)

Search outputs

1

Key	Value	Description	Export name
BatchTargetBucket	adp-or-batch-4d10a160-0060-11ee-af5f-0abeeafc31c1	Target bucket for Batch usecase.	-
DataCaptureBucket	sagemaker-adp-or-us-west-2-	Data capture bucket for ml observability	-
EstimatorBucket	adp-or-us-west-2-estimator-	Estimator bucket for ml observability	-
MlObservabilityNotebookInstanceName	adp-observability-instance	SageMaker notebook instance name for Observability	-
MlObservabilityNotebookURL	https://us-west-2.console.aws.amazon.com/sagemaker/home?region=us-west-2#/notebook-instances/openNotebook/adp-observability-instance	SageMaker Notebook Instance URL for Promotion Planning case	-

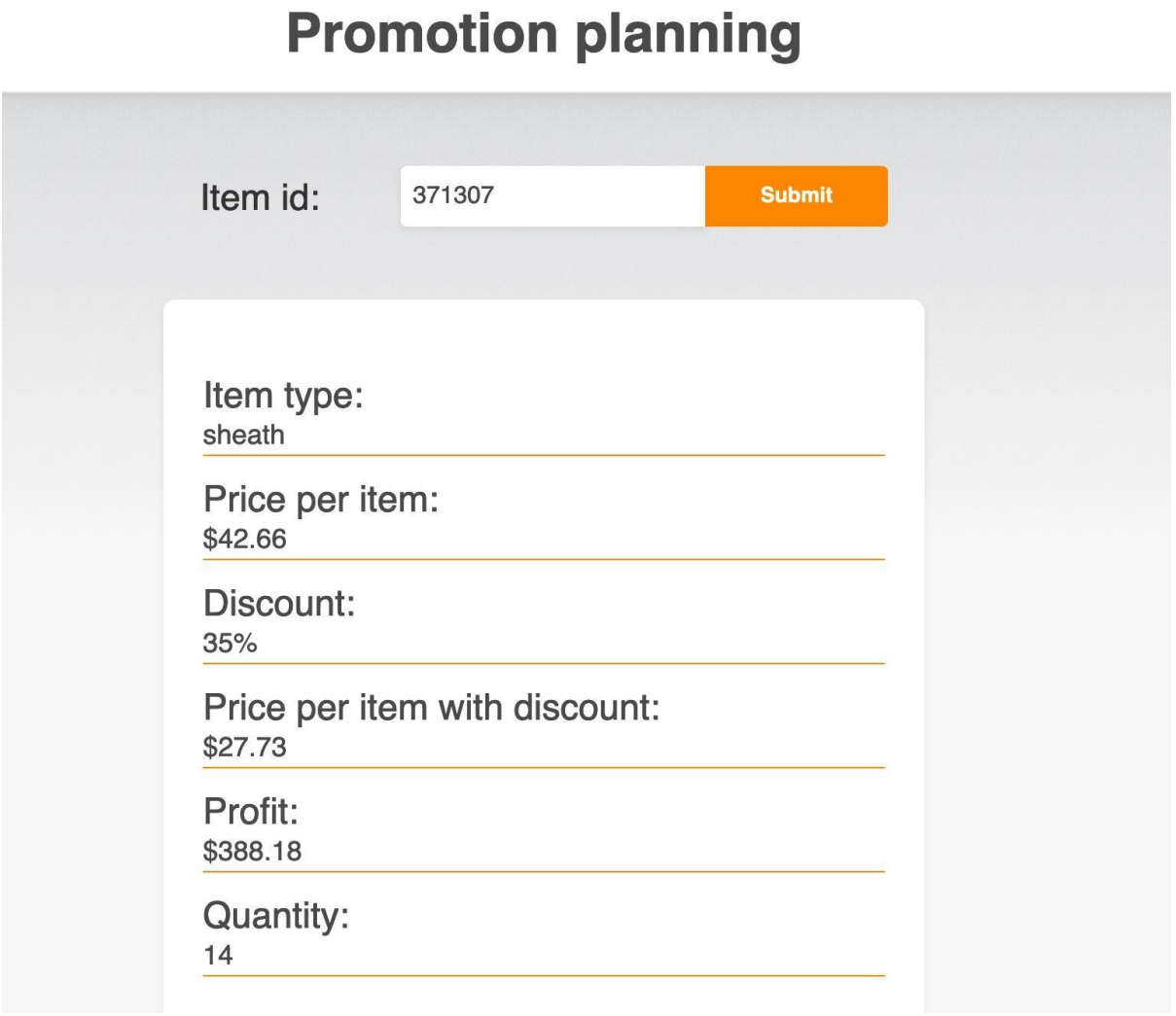
Important!

1. Before you proceed with cloudformation template installation, pull the image **NannymlDockerImage** from Marketplace, create ECR in your AWS account and push it there. Change the path to yours ECR in **NannymlDockerImage** when launch/update use cases.

Promotion planning

Enabling MLOps with the data platform and providing an easy way to release models to production is always a cumbersome process. To simplify ML models development and management, it is helpful to provide proper integration with the data lake and enterprise data warehouse.

There are several use cases we've added on top of the ML platform for demo purposes. Promotion planning is one of these. The promotion planning use case creates the ML model, which creates the discount recommendations.



Recommendations are based on sales history information available in the data platform.

Promotion planning provisions Sagemaker, creates notebook instances, and loads an already prepared Jupyter notebook. Data required for the model is available at S3 bucket adp-rnd-ml-datasets prepared by our team. The model will be stored in a private bucket that is provisioned by CloudFormation automation. Once a model is created, it also spins up a web UI to demonstrate the recommendation in action. Capability provisioning is straightforward:

General configurations for cases Promotion Planning, Visual Attributes, MlOps

NotebookInstanceType

The EC2 instance type for the data lake Amazon SageMaker Notebook instance.

ml.t3.large

Default value can be overridden.

MlModelDockerImage

Docker image name for Ml Model service

griddynamics/adp-model-deployment

Default value can be overridden.

MlflowServiceDockerImage

Enter your path of image. Repository should be created before starting this cases

griddynamics/adp-mlflow-churn-predictions

Default value can be overridden.

MlflowExperimentDockerImage

Docker image name for MlflowExperiment lambda

griddynamics/mlflowexperiment

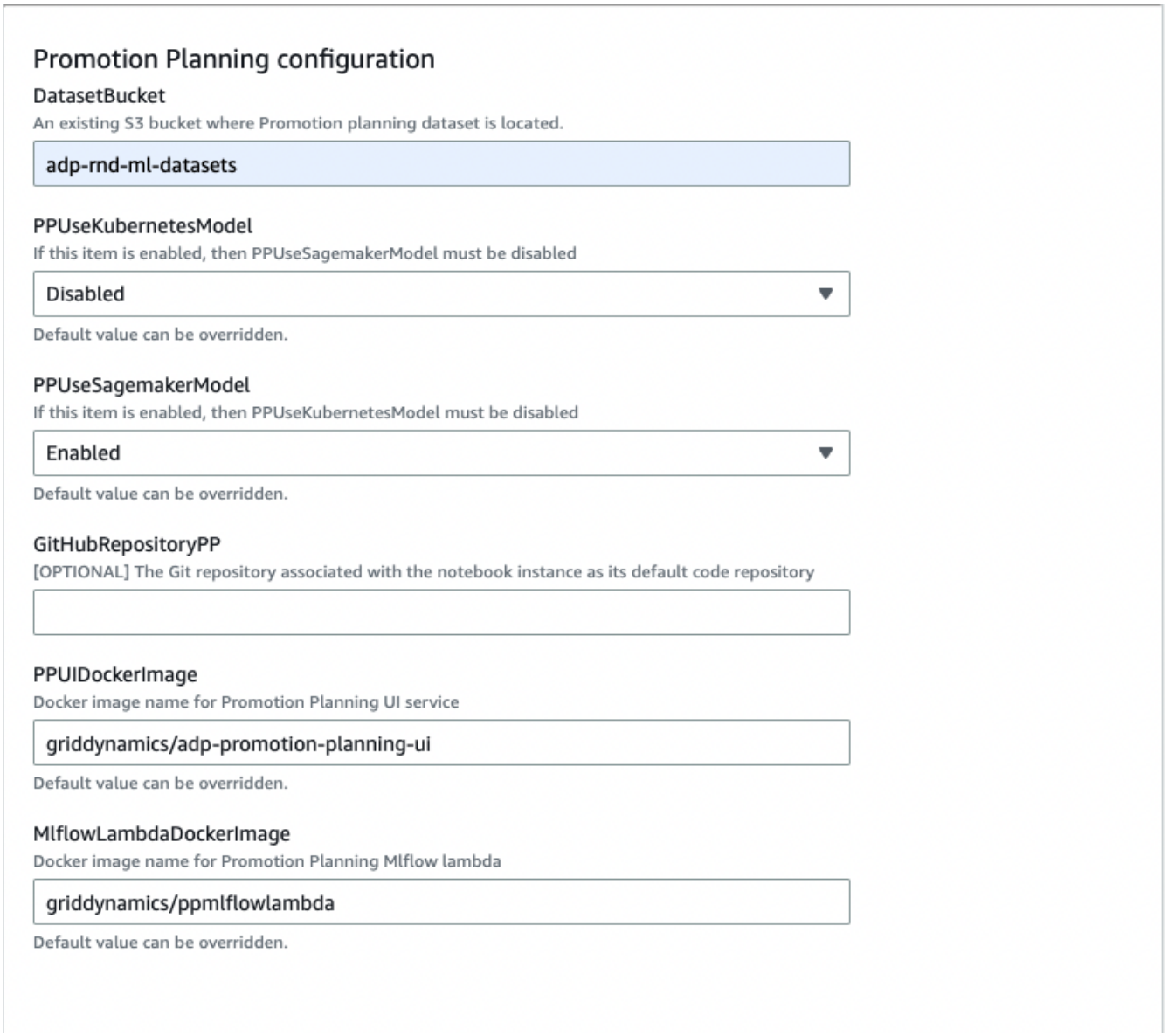
Default value can be overridden.

CleanupEpLambdaDockerImage

Docker image name for Cleanup Sagemaker Endpoint Lambda

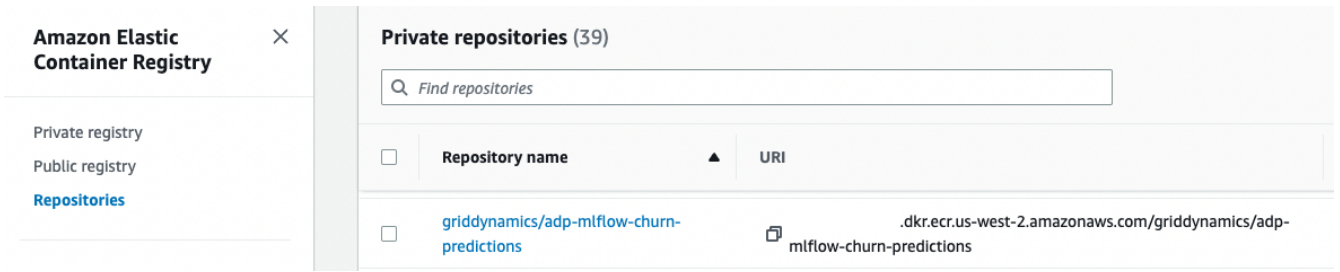
griddynamics/cleanup-ep-lambda

Default value can be overridden.



Important!

1. Before you proceed with cloudformation template installation, pull the images (**MlflowExperimentDockerImage**, **CleanupEpLambdaDockerImage**, **MlflowLambdaDockerImage**) from Marketplace, create ECR in your AWS account and push it there. Change the path to yours ECR in MlflowExperimentDockerImage, CleanupEpLambdaDockerImage, MlflowLambdaDockerImage when launch/update use cases
2. Before running this case **MlflowServiceImageRepository** should be created at AWS Account on Elastic Container Registry Resource. It's can be looks like this:

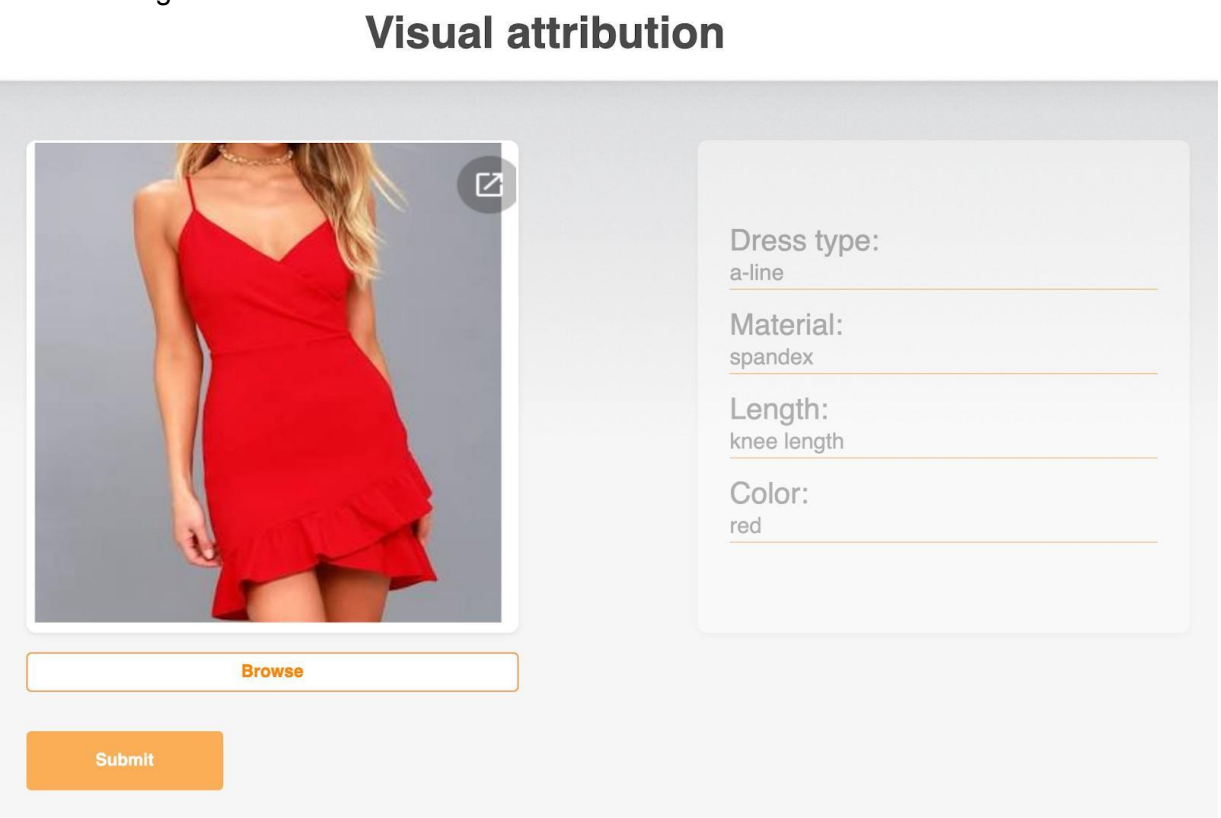


All variables are defined by CloudFormation automation. By default all components are not accessible outside a VPN. However, if needed access can be granted for the world wide

internet, which can be done enable EnableExternalAccess in use case configuration and base platform

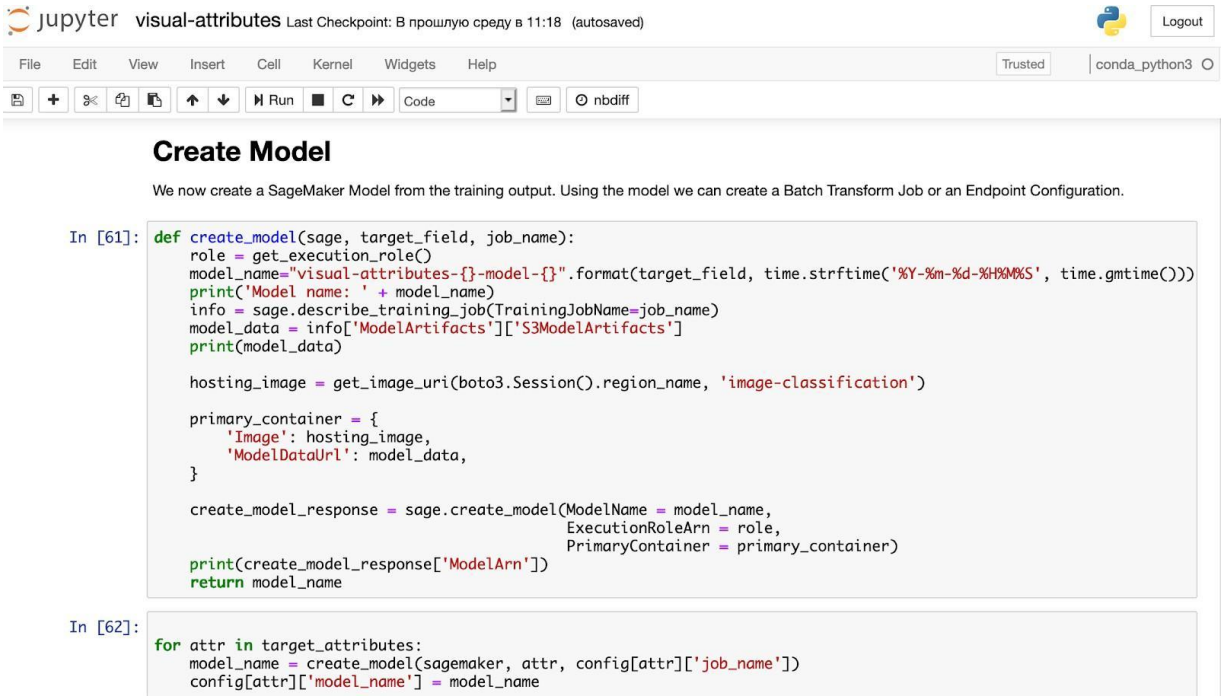
Visual attributes

The data platform can typically be operated with all types of data including transactional, operational, and visual data such as images. The visual attributes use case gets attributes from the image:

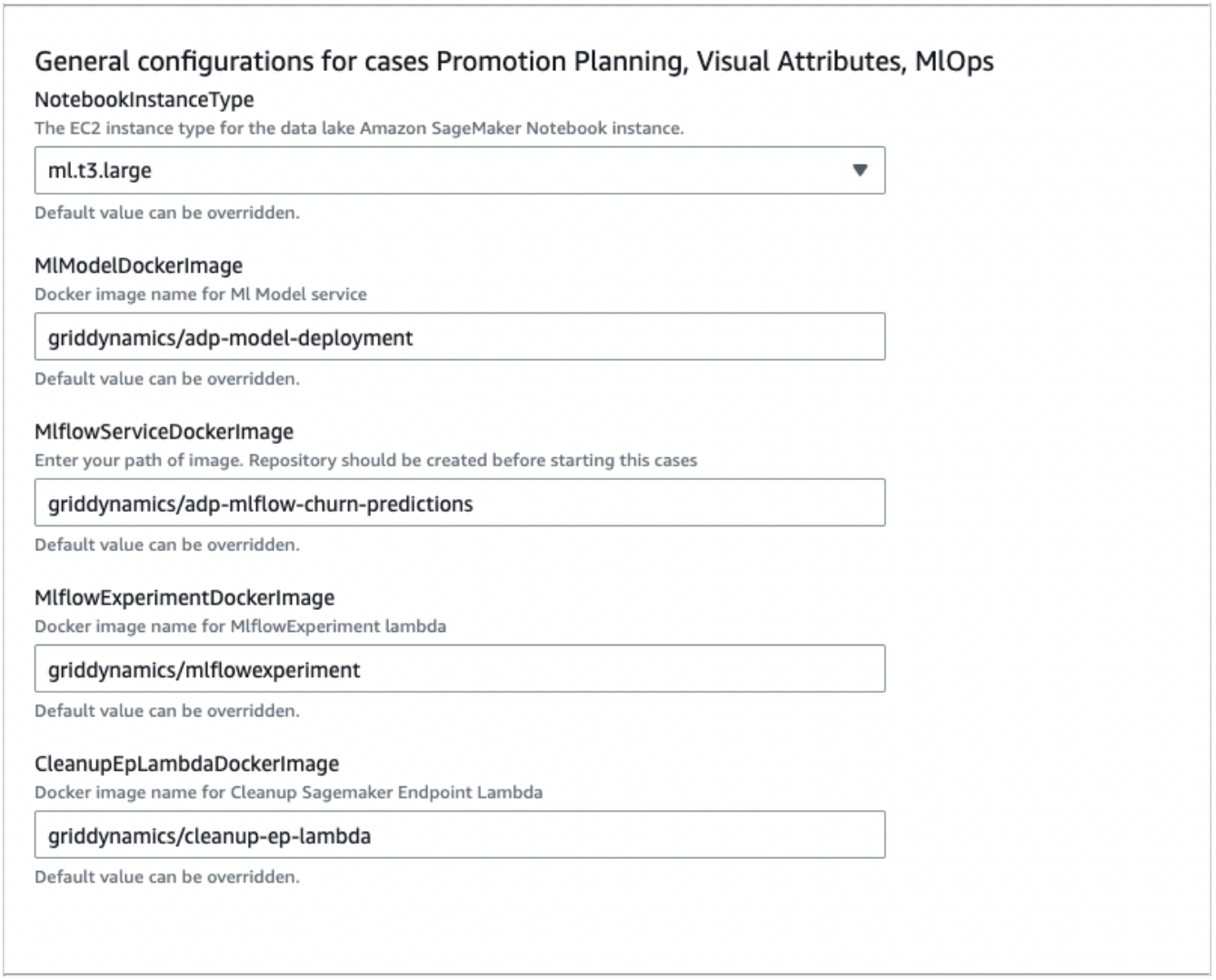


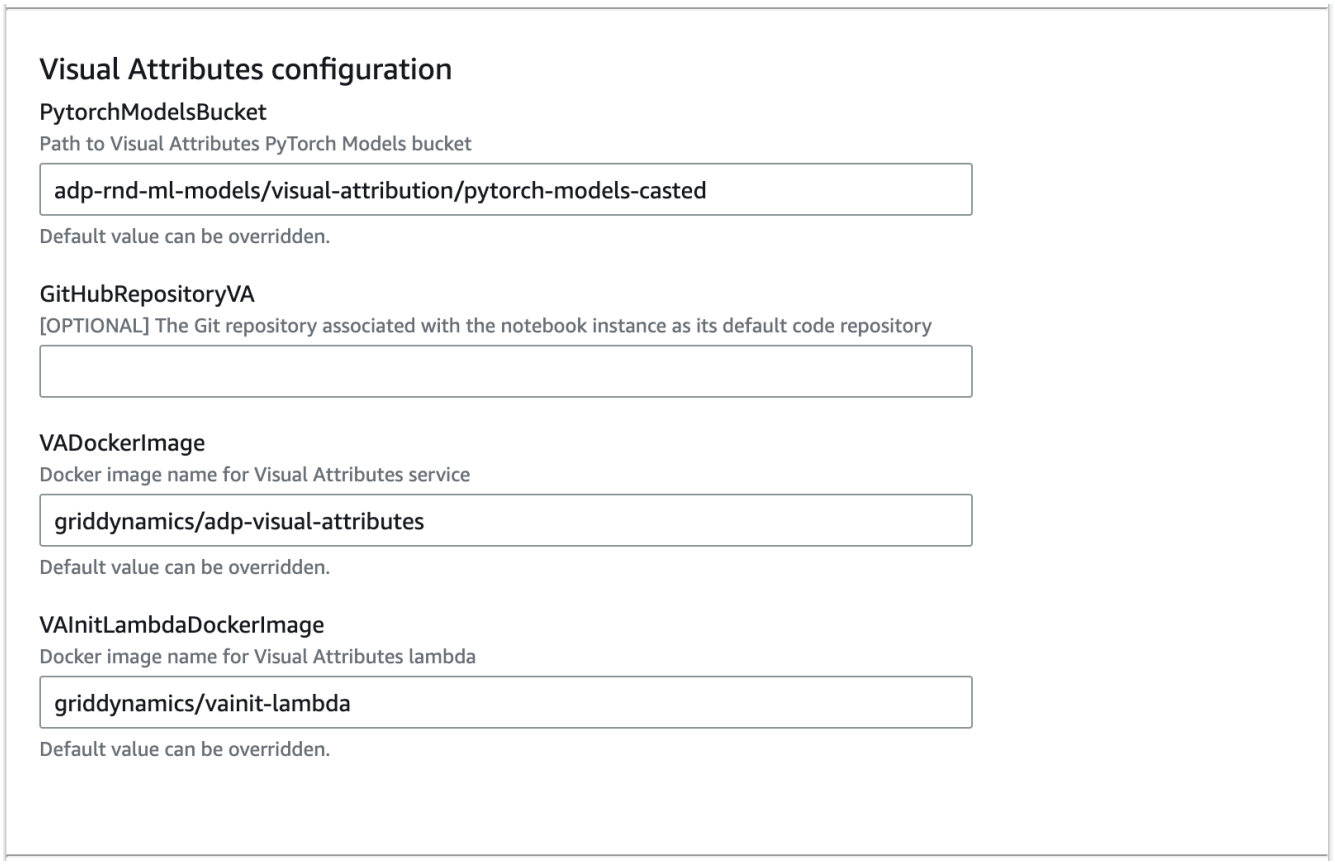
As an example we've taken a catalog of women's dresses and created a use case that identifies the type and color of the dress and helps to find a particular one in the catalog. Similar to the promotion planning use case above, links to both private and public links to the user interface will be available in the provisioned product details.

Source images for the model are also in the adp-rnd-ml-datasets S3 bucket. Jupyter notebook reads data directly and creates both the ML model and REST endpoint for the UI:



Visual attributes in Jupyter notebook are also created by CloudFormation automation. The ML engineer will just need to run the training and release process. Both visual attribute and promotion planning use cases help to understand and build the custom ML pipelines. Visual attributes provisioning works the same way:

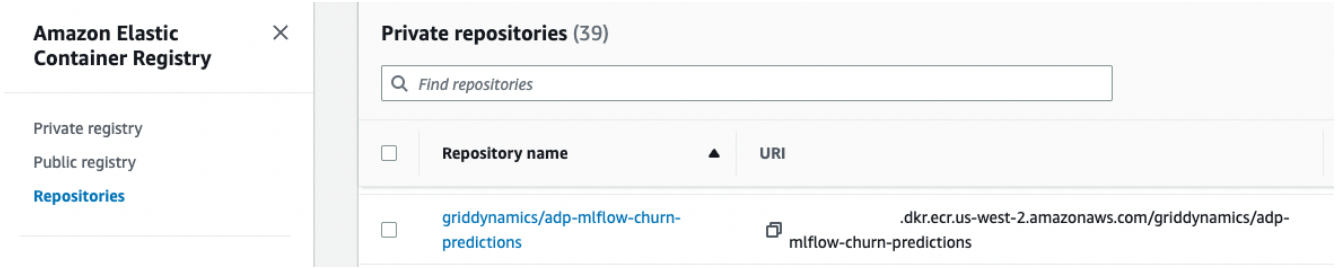




The visual attributes configuration process is the same as the promotion planning use case discussed above.

Important!

1. Before you proceed with cloudformation template installation, pull the images (**MlflowExperimentDockerImage**, **CleanupEpLambdaDockerImage**, **VAinitLambdaDockerImage**) from Marketplace, create ECR in your AWS account and push it there. Change the path to yours ECR in **MlflowExperimentDockerImage**, **CleanupEpLambdaDockerImage**, **VAinitLambdaDockerImage** when launch/update use cases
2. Before running this case **MlflowServiceImageRepository** should be created at AWS Account on Elastic Container Registry Resource. It's can be looks like this:

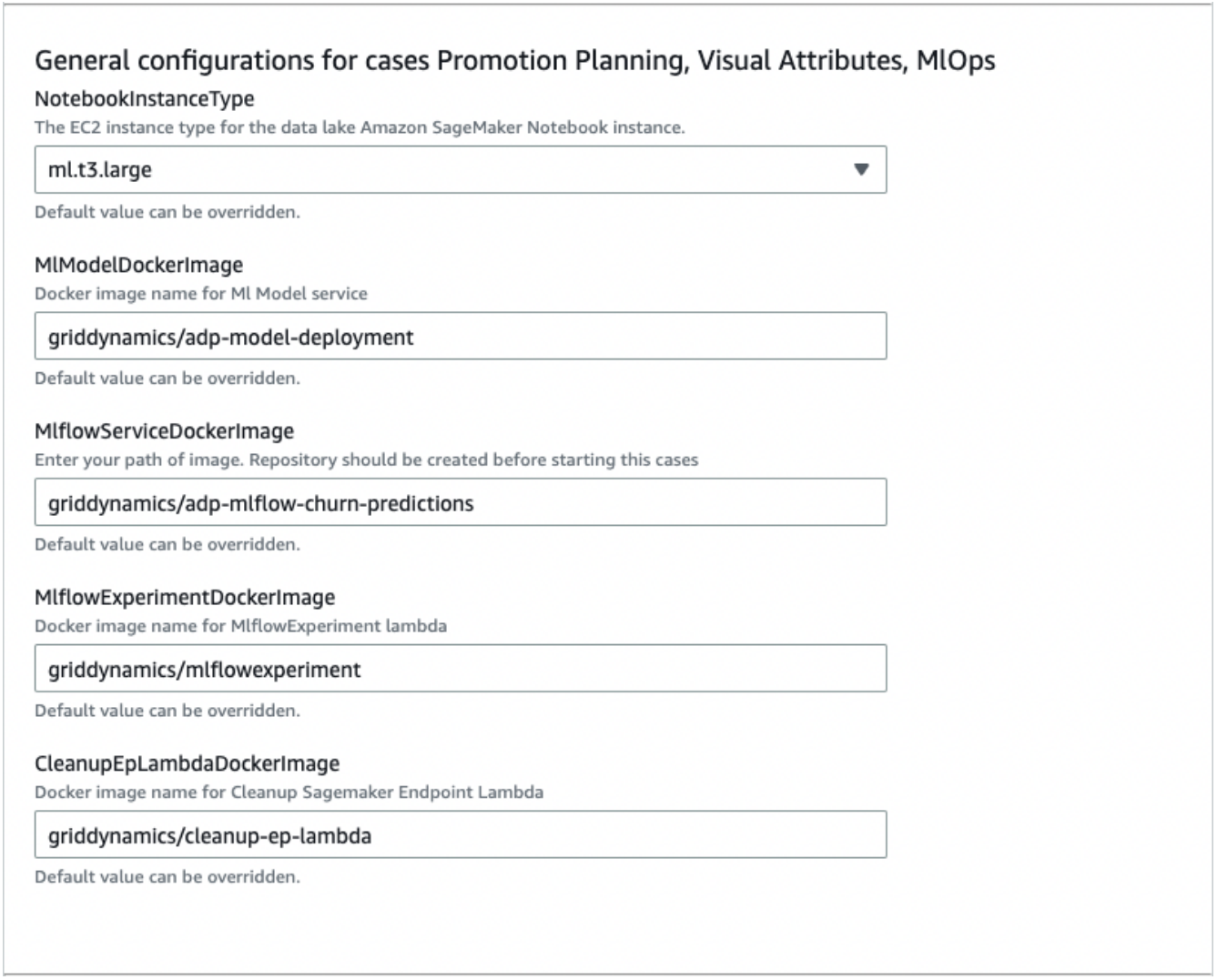


MLOps

ML platform is infrastructure-as-a-code solution which provides tooling and automation to develop, test, pass though CI pipeline, create artifacts and release as REST endpoint. Platform support two ways of serving: through standard SageMaker REST endpoints or running on top of Kubernetes.

MLOps provisions Sagemaker, creates notebook instances, and loads an already prepared Jupyter notebook. Data required for the model is available at S3 bucket adp-rnd-ml-datasets prepared by our team. The model will be stored in a private bucket

that is provisioned by CloudFormation automation. Once a model is created, it also spins up a web UI to demonstrate the recommendation in action. Capability provisioning is straightforward:



MlOps configuration

MlOpsUseKubernetesModel
[WARNING] If this item is enabled, then MlOpsUseSagemakerModel must be disabled

Disabled

Default value can be overridden.

MlOpsUseSagemakerModel
[WARNING] If this item is enabled, then MlOpsUseKubernetesModel must be disabled

Enabled

Default value can be overridden.

DeployUsingCodeBuild
[WARNING] Should match with DeployUsingLambda parameter. If parameter DeployUsingLambda was enabled, then this parameter must be disabled!

Disabled

Default value can be overridden.

DeployUsingLambda
[WARNING] Should match with DeployUsingCodeBuild parameter. If parameter DeployUsingCodeBuild was enabled, then this parameter must be disabled!

Enabled

Default value can be overridden.

GitHubRepositoryMlOps
[OPTIONAL] The Git repository associated with the notebook instance as its default code repository

CPDockerImage
Docker image name for Churn Prediction service

griddynamics/adp-churn-predictions

Default value can be overridden.

MlLambdaDockerImage
Docker image name for Ml lambda

griddynamics/mllambda

Default value can be overridden.

RepositoryID
[If DeployUsingCodeBuild enabled] The owner and name of the repository where source changes are to be detected. For example, griddynamics/test

BranchName
[If DeployUsingCodeBuild enabled] The name of the branch for the repository where the source change was made

main

Default value can be overridden.

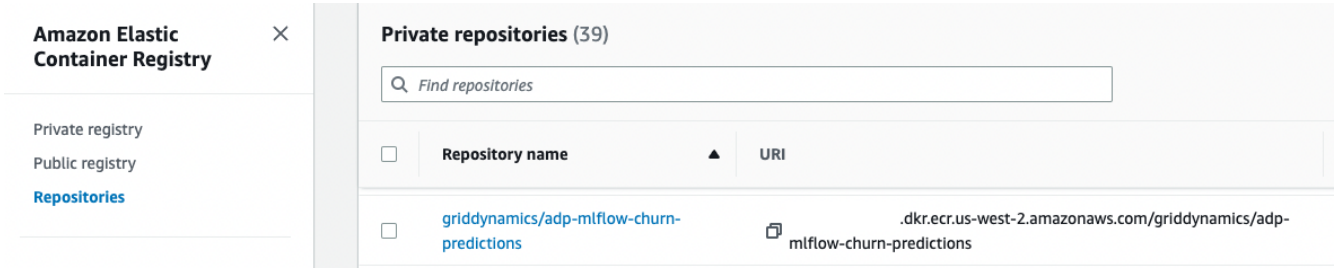
GitConnectionName
[If DeployUsingCodeBuild enabled] The name of the connection. Connection names must be unique in an AWS user account. MaxLength of name - 32.

MlflowGHConnection

Default value can be overridden.

Important!

1. Before you proceed with cloudformation template installation, pull the images (**MlflowExperimentDockerImage**, **CleanupEpLambdaDockerImage**, **MlLambdaDockerImage**) from Marketplace, create ECR in your AWS account and push it there. Change the path to yours ECR in **MlflowExperimentDockerImage**, **CleanupEpLambdaDockerImage**, **MlLambdaDockerImage** when launch/update use cases
2. Before running this case **MlflowServiceImageRepository** should be created at AWS Account on Elastic Container Registry Resource. It's can be looks like this:



Anomaly detection

Anomaly detection capability is fully covered in a separate post.

Conclusion

Analytical Data Platform is a modular platform with six major capabilities. Each of these can be deployed separately or share the same resources like EMR or Redshift. Deployment is based on the CloudFormation stack and is fully automated. It takes less than a day to deploy all capabilities however, for promotion planning and visual attributes it takes a significant amount of time to train and create the models. The platform is fully operable and ready to be integrated with external data sources.