» mongodbatlas_database_user

`mongodbatlas_database_user` describe a Database User. This represents a database user which will be applied to all clusters within the project.

Each user has a set of roles that provide access to the project's databases. User's roles apply to all the clusters in the project: if two clusters have a `products` database and a user has a role granting `read` access on the products database, the user has that access on both clusters.

NOTE: Groups and projects are synonymous terms. You may find group_id in the official documentation.

» Example Usage

```
resource "mongodbatlas_database_user" "test" {
  username           = "test-acc-username"
  password           = "test-acc-password"
  project_id         = "<PROJECT-ID>"
  auth_database_name = "admin"

  roles {
    role_name     = "readWrite"
    database_name = "admin"
  }

  roles {
    role_name     = "atlasAdmin"
    database_name = "admin"
  }

  labels {
    key   = "key 1"
    value = "value 1"
  }
  labels {
    key   = "key 2"
    value = "value 2"
  }
}

data "mongodbatlas_database_user" "test" {
  project_id = mongodbatlas_database_user.test.project_id
  username   = mongodbatlas_database_user.test.username
}
```

» Argument Reference

- `username` - (Required) Username for authenticating to MongoDB.
- `project_id` - (Required) The unique ID for the project to create the database user.
- `auth_database_name` - (Required) The user's authentication database. A user must provide both a username and authentication database to log into MongoDB. In Atlas deployments of MongoDB, the authentication database is almost always the admin database, for X509 it is $external.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The database user's name.
- `roles` - List of user's roles and the databases / collections on which the roles apply. A role allows the user to perform particular actions on the specified database. A role on the admin database can include privileges that apply to the other databases as well. See Roles below for more details.
- `x509_type` - X.509 method by which the provided username is authenticated.

» Roles

Block mapping a user's role to a database / collection. A role allows the user to perform particular actions on the specified database. A role on the admin database can include privileges that apply to the other databases as well.

NOTE: The available privilege actions for custom MongoDB roles support a subset of MongoDB commands. See Unsupported Commands in M10+ Clusters for more information.

IMPORTANT: If a user is assigned a custom MongoDB role, they cannot be assigned any other roles.

- `name` - Name of the role to grant.
- `database_name` - Database on which the user has the specified role. A role on the `admin` database can include privileges that apply to the other databases.
- `collection_name` - Collection for which the role applies. You can specify a collection for the `read` and `readWrite` roles. If you do not specify a collection for `read` and `readWrite`, the role applies to all collections in the database (excluding some collections in the `system`. database).

» Labels

Containing key-value pairs that tag and categorize the database user. Each key and value has a maximum length of 255 characters.

- `key` - The key that you want to write.
- `value` - The value that you want to write.

See MongoDB Atlas API Documentation for more information.

» mongodbatlas_database_users

`mongodbatlas_database_users` describe all Database Users. This represents a database user which will be applied to all clusters within the project.

Each user has a set of roles that provide access to the project's databases. User's roles apply to all the clusters in the project: if two clusters have a `products` database and a user has a role granting `read` access on the products database, the user has that access on both clusters.

NOTE: Groups and projects are synonymous terms. You may find `groupId` in the official documentation.

» Example Usage

```
resource "mongodbatlas_database_user" "test" {
  username           = "test-acc-username"
  password           = "test-acc-password"
  project_id         = "<PROJECT-ID>"
  auth_database_name = "admin"

  roles {
    role_name     = "readWrite"
    database_name = "admin"
  }

  roles {
    role_name     = "atlasAdmin"
    database_name = "admin"
  }

  labels {
    key   = "key 1"
    value = "value 1"
  }
```

```
  labels {
    key   = "key 2"
    value = "value 2"
  }
}

data "mongodbatlas_database_users" "test" {
  project_id = mongodbatlas_database_user.test.project_id
}
```

» Argument Reference

- `project_id` - (Required) The unique ID for the project to get all database users.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - Autogenerated Unique ID for this data source.
- `results` - A list where each represents a Database user.

» Database User

- `project_id` - ID of the Atlas project the user belongs to.
- `username` - Username for authenticating to MongoDB.
- `roles` - List of user's roles and the databases / collections on which the roles apply. A role allows the user to perform particular actions on the specified database. A role on the admin database can include privileges that apply to the other databases as well. See Roles below for more details.
- `auth_database_name` - The user's authentication database. A user must provide both a username and authentication database to log into MongoDB. In Atlas deployments of MongoDB, the authentication database is always the admin database.
- `x509_type` - X.509 method by which the provided username is authenticated.

» Roles

Block mapping a user's role to a database / collection. A role allows the user to perform particular actions on the specified database. A role on the admin

database can include privileges that apply to the other databases as well.

NOTE: The available privilege actions for custom MongoDB roles support a subset of MongoDB commands. See Unsupported Commands in M10+ Clusters for more information.

IMPORTANT: If a user is assigned a custom MongoDB role, they cannot be assigned any other roles.

- `name` - Name of the role to grant.
- `database_name` - Database on which the user has the specified role. A role on the `admin` database can include privileges that apply to the other databases.
- `collection_name` - Collection for which the role applies. You can specify a collection for the `read` and `readWrite` roles. If you do not specify a collection for `read` and `readWrite`, the role applies to all collections in the database (excluding some collections in the `system`. database).

» Labels

Containing key-value pairs that tag and categorize the database user. Each key and value has a maximum length of 255 characters.

- `key` - The key that you want to write.
- `value` - The value that you want to write.

See MongoDB Atlas API Documentation for more information.

» mongodbatlas_project

`mongodbatlas_project` describes a MongoDB Atlas Project. This represents a project that has been created.

NOTE: Groups and projects are synonymous terms. You may find group_id in the official documentation.

» Example Usage

» Using project_id attribute to query

```
resource "mongodbatlas_project" "test" {
  name   = "project-name"
  org_id = "<ORG_ID>"

  teams {
```

```
    team_id    = "5e0fa8c99ccf641c722fe645"
    role_names = ["GROUP_OWNER"]

  }
  teams {
    team_id    = "5e1dd7b4f2a30ba80a70cd4rw"
    role_names = ["GROUP_READ_ONLY", "GROUP_DATA_ACCESS_READ_WRITE"]
  }
}

data "mongodbatlas_project" "test" {
  project_id = "${mongodbatlas_project.test.id}"
}
```

» Using name attribute to query

```
resource "mongodbatlas_project" "test" {
  name   = "project-name"
  org_id = "<ORG_ID>"

  teams {
    team_id    = "5e0fa8c99ccf641c722fe645"
    role_names = ["GROUP_OWNER"]

  }
  teams {
    team_id    = "5e1dd7b4f2a30ba80a70cd4rw"
    role_names = ["GROUP_READ_ONLY", "GROUP_DATA_ACCESS_READ_WRITE"]
  }
}

data "mongodbatlas_project" "test" {
  name = "${mongodbatlas_project.test.name}"
}
```

» Argument Reference

- `project_id` - (Optional) The unique ID for the project.
- `name` - (Optional) The unique ID for the project.

IMPORTANT: Either `project_id` or `name` must be configurated.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `name` - The name of the project you want to create. (Cannot be changed via this Provider after creation.)
- `org_id` - The ID of the organization you want to create the project within. *`cluster_count` - The number of Atlas clusters deployed in the project. *`created` - The ISO-8601-formatted timestamp of when Atlas created the project.
- `teams.#.team_id` - The unique identifier of the team you want to associate with the project. The team and project must share the same parent organization.
- `teams.#.role_names` - Each string in the array represents a project role assigned to the team. Every user associated with the team inherits these roles. The following are valid roles:
  - `GROUP_OWNER`
  - `GROUP_READ_ONLY`
  - `GROUP_DATA_ACCESS_ADMIN`
  - `GROUP_DATA_ACCESS_READ_WRITE`
  - `GROUP_DATA_ACCESS_READ_ONLY`
  - `GROUP_CLUSTER_MANAGER`

See MongoDB Atlas API - Project - and MongoDB Atlas API - Teams Documentation for more information.

» mongodbatlas_projects

`mongodbatlas_projects` describe all Projects. This represents projects that have been created.

NOTE: Groups and projects are synonymous terms. You may find `groupId` in the official documentation.

» Example Usage

```
resource "mongodbatlas_project" "test" {
  name   = "project-name"
  org_id = "<ORG_ID>"

  teams {
    team_id    = "5e0fa8c99ccf641c722fe645"
    role_names = ["GROUP_OWNER"]

  }
```

```
  teams {
    team_id    = "5e1dd7b4f2a30ba80a70cd4rw"
    role_names = ["GROUP_READ_ONLY", "GROUP_DATA_ACCESS_READ_WRITE"]
  }
}

data "mongodbatlas_project" "test" {}
```

» Argument Reference

There is not arguments

- `id` - Autogenerated Unique ID for this data source.
- `total_count` - Represents the total of the projects
- `results` - A list where each represents a Projects.

» Project

- `name` - The name of the project you want to create. (Cannot be changed via this Provider after creation.)
- `org_id` - The ID of the organization you want to create the project within. *`cluster_count` - The number of Atlas clusters deployed in the project. *`created` - The ISO-8601-formatted timestamp of when Atlas created the project.
- `teams.#.team_id` - The unique identifier of the team you want to associate with the project. The team and project must share the same parent organization.
- `teams.#.role_names` - Each string in the array represents a project role assigned to the team. Every user associated with the team inherits these roles. The following are valid roles:
  - `GROUP_OWNER`
  - `GROUP_READ_ONLY`
  - `GROUP_DATA_ACCESS_ADMIN`
  - `GROUP_DATA_ACCESS_READ_WRITE`
  - `GROUP_DATA_ACCESS_READ_ONLY`
  - `GROUP_CLUSTER_MANAGER`

See MongoDB Atlas API - Projects - and MongoDB Atlas API - Teams Documentation for more information.

» mongodbatlas_cluster

`mongodbatlas_cluster` describes a Cluster. The. The data source requires your Project ID.

NOTE: Groups and projects are synonymous terms. You may find group_id in the official documentation.

IMPORTANT:
• Changes to cluster configurations can affect costs. Before making changes, please see Billing.
• If your Atlas project contains a custom role that uses actions introduced in a specific MongoDB version, you cannot create a cluster with a MongoDB version less than that version unless you delete the custom role.

» Example Usage

```
resource "mongodbatlas_cluster" "test" {
  project_id   = "<YOUR-PROJECT-ID>"
  name         = "cluster-test"
  disk_size_gb = 100
  num_shards   = 1

  replication_factor           = 3
  backup_enabled               = true
  auto_scaling_disk_gb_enabled = true

  //Provider Settings "block"
  provider_name               = "AWS"
  provider_disk_iops          = 300
  provider_volume_type        = "STANDARD"
  provider_encrypt_ebs_volume = true
  provider_instance_size_name = "M40"
  provider_region_name        = "US_EAST_1"
}

data "mongodbatlas_cluster" "test" {
    project_id = mongodbatlas_cluster.test.project_id
    name       = mongodbatlas_cluster.test.name
}
```

» Argument Reference

- `project_id` - (Required) The unique ID for the project to create the database user.

- `name` - (Required) Name of the cluster as it appears in Atlas. Once the cluster is created, its name cannot be changed.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The cluster ID.
- `mongo_db_version` - Version of MongoDB the cluster runs, in `major-version.minor-version` format.
- `mongo_uri` - Base connection string for the cluster. Atlas only displays this field after the cluster is operational, not while it builds the cluster.
- `mongo_uri_updated` - Lists when the connection string was last updated. The connection string changes, for example, if you change a replica set to a sharded cluster.
- `mongo_uri_with_options` - Describes connection string for connecting to the Atlas cluster. Includes the replicaSet, ssl, and authSource query parameters in the connection string with values appropriate for the cluster.

  To review the connection string format, see the connection string format documentation. To add MongoDB users to a Atlas project, see Configure MongoDB Users.

  Atlas only displays this field after the cluster is operational, not while it builds the cluster.
- `paused` - Flag that indicates whether the cluster is paused or not.
- `pit_enabled` - Flag that indicates if the cluster uses Point-in-Time backups.
- `srv_address` - Connection string for connecting to the Atlas cluster. The +srv modifier forces the connection to use TLS/SSL. See the mongoURI for additional options.
- `state_name` - Indicates the current state of the cluster. The possible states are:
  - IDLE
  - CREATING
  - UPDATING
  - DELETING
  - DELETED
  - REPAIRING
- `auto_scaling_disk_gb_enabled` - Indicates whether disk auto-scaling is enabled.

- `backup_enabled` - Indicates whether Atlas continuous backups are enabled for the cluster.
- `bi_connector` - Indicates BI Connector for Atlas configuration on this cluster. BI Connector for Atlas is only available for M10+ clusters. See BI Connector below for more details.
- `cluster_type` - Indicates the type of the cluster that you want to modify. You cannot convert a sharded cluster deployment to a replica set deployment.
- `disk_size_gb` - Indicates the size in gigabytes of the server's root volume (AWS/GCP Only).
- `encryption_at_rest_provider` - Indicates whether Encryption at Rest is enabled or disabled.
- `name` - Name of the cluster as it appears in Atlas.
- `mongo_db_major_version` - Indicates the version of the cluster to deploy.
- `num_shards` - Indicates whether the cluster is a replica set or a sharded cluster.
- `provider_backup_enabled` - Flag indicating if the cluster uses Cloud Provider Snapshots for backups.
- `provider_instance_size_name` - Atlas provides different instance sizes, each with a default storage capacity and RAM size.
- `provider_name` - Indicates the cloud service provider on which the servers are provisioned.
- `backing_provider_name` - Indicates Cloud service provider on which the server for a multi-tenant cluster is provisioned.
- `provider_disk_iops` - Indicates the maximum input/output operations per second (IOPS) the system can perform. The possible values depend on the selected providerSettings.instanceSizeName and diskSizeGB.
- `provider_disk_type_name` - Describes Azure disk type of the server's root volume (Azure Only).
- `provider_encrypt_ebs_volume` - Indicates whether the Amazon EBS encryption is enabled. This feature encrypts the server's root volume for both data at rest within the volume and data moving between the volume and the instance.
- `provider_region_name` - Indicates Physical location of your MongoDB cluster. The region you choose can affect network latency for clients accessing your databases. Requires the Atlas Region name, see the reference list for AWS, GCP, Azure.

- `provider_volume_type` - Indicates the type of the volume. The possible values are: `STANDARD` and `PROVISIONED`.
- `replication_factor` - Number of replica set members. Each member keeps a copy of your databases, providing high availability and data redundancy. The possible values are 3, 5, or 7. The default value is 3.
- `replication_specs` - Configuration for cluster regions. See Replication Spec below for more details.

» BI Connector

Indicates BI Connector for Atlas configuration.

- `enabled` - Indicates whether or not BI Connector for Atlas is enabled on the cluster.
- `read_preference` - Indicates the read preference to be used by BI Connector for Atlas on the cluster. Each BI Connector for Atlas read preference contains a distinct combination of readPreference and readPreferenceTags options. For details on BI Connector for Atlas read preferences, refer to the BI Connector Read Preferences Table.

» Replication Spec

Configuration for cluster regions.

- `id` - Unique identifer of the replication document for a zone in a Global Cluster.
- `num_shards` - Number of shards to deploy in the specified zone.
- `regions_config` - Describes the physical location of the region. Each regionsConfig document describes the region's priority in elections and the number and type of MongoDB nodes Atlas deploys to the region. You must order each regionsConfigs document by regionsConfig.priority, descending. See Region Config below for more details.
- `zone_name` - Indicates the n ame for the zone in a Global Cluster.

» Region Config

Physical location of the region.

- `region_name` - Name for the region specified.
- `electable_nodes` - Number of electable nodes for Atlas to deploy to the region.
- `priority` - Election priority of the region. For regions with only read-only nodes, set this value to 0.

- `read_only_nodes` - Number of read-only nodes for Atlas to deploy to the region. Read-only nodes can never become the primary, but can facilitate local-reads. Specify 0 if you do not want any read-only nodes in the region.
- `analytics_nodes` - Indicates the number of analytics nodes for Atlas to deploy to the region. Analytics nodes are useful for handling analytic data such as reporting queries from BI Connector for Atlas. Analytics nodes are read-only, and can never become the primary.

» Labels

Contains key-value pairs that tag and categorize the cluster. Each key and value has a maximum length of 255 characters.

- `key` - The key that was set.
- `value` - The value that represents the key.

» Plugin

Contains a key-value pair that tags that the cluster was created by a Terraform Provider and notes the version.

- `name` - The name of the current plugin
- `version` - The current version of the plugin.

See detailed information for arguments and attributes: MongoDB API Clusters

» mongodbatlas_clusters

`mongodbatlas_cluster` describes all Clusters by the provided project_id. The data source requires your Project ID.

NOTE: Groups and projects are synonymous terms. You may find group_id in the official documentation.

IMPORTANT:
• Changes to cluster configurations can affect costs. Before making changes, please see Billing.
• If your Atlas project contains a custom role that uses actions introduced in a specific MongoDB version, you cannot create a cluster with a MongoDB version less than that version unless you delete the custom role.

» Example Usage

```
resource "mongodbatlas_cluster" "test" {
```

```
  project_id   = "<YOUR-PROJECT-ID>"
  name         = "cluster-test"
  disk_size_gb = 100
  num_shards   = 1

  replication_factor           = 3
  backup_enabled               = true
  auto_scaling_disk_gb_enabled = true

  //Provider Settings "block"
  provider_name               = "AWS"
  provider_disk_iops          = 300
  provider_volume_type        = "STANDARD"
  provider_encrypt_ebs_volume = true
  provider_instance_size_name = "M40"
  provider_region_name        = "US_EAST_1"
}

data "mongodbatlas_clusters" "test" {
    project_id = mongodbatlas_cluster.test.project_id // To get dependency.
}
```

» Argument Reference

- `project_id` - (Required) The unique ID for the project to get the clusters.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The cluster ID.
- `results` - A list where each represents a Cluster. See Cluster below for more details.

» Cluster

- `name` - Name of the cluster as it appears in Atlas.
- `mongo_db_version` - Version of MongoDB the cluster runs, in `major-version.minor-version` format.
- `mongo_uri` - Base connection string for the cluster. Atlas only displays this field after the cluster is operational, not while it builds the cluster.

- `mongo_uri_updated` - Lists when the connection string was last updated. The connection string changes, for example, if you change a replica set to a sharded cluster.
- `mongo_uri_with_options` - Describes connection string for connecting to the Atlas cluster. Includes the replicaSet, ssl, and authSource query parameters in the connection string with values appropriate for the cluster.

  To review the connection string format, see the connection string format documentation. To add MongoDB users to a Atlas project, see Configure MongoDB Users.

  Atlas only displays this field after the cluster is operational, not while it builds the cluster.
- `paused` - Flag that indicates whether the cluster is paused or not.
- `pit_enabled` - Flag that indicates if the cluster uses Point-in-Time backups.
- `srv_address` - Connection string for connecting to the Atlas cluster. The +srv modifier forces the connection to use TLS/SSL. See the mongoURI for additional options.
- `state_name` - Indicates the current state of the cluster. The possible states are:
  - IDLE
  - CREATING
  - UPDATING
  - DELETING
  - DELETED
  - REPAIRING
- `auto_scaling_disk_gb_enabled` - Indicates whether disk auto-scaling is enabled.
- `backup_enabled` - Indicates whether Atlas continuous backups are enabled for the cluster.
- `bi_connector` - Indicates BI Connector for Atlas configuration on this cluster. BI Connector for Atlas is only available for M10+ clusters. See BI Connector below for more details.
- `cluster_type` - Indicates the type of the cluster that you want to modify. You cannot convert a sharded cluster deployment to a replica set deployment.
- `disk_size_gb` - Indicates the size in gigabytes of the server's root volume (AWS/GCP Only).
- `encryption_at_rest_provider` - Indicates whether Encryption at Rest is enabled or disabled.

- `mongo_db_major_version` - Indicates the version of the cluster to deploy.
- `num_shards` - Indicates whether the cluster is a replica set or a sharded cluster.
- `provider_backup_enabled` - Flag indicating if the cluster uses Cloud Provider Snapshots for backups.
- `provider_instance_size_name` - Atlas provides different instance sizes, each with a default storage capacity and RAM size.
- `provider_name` - Indicates the cloud service provider on which the servers are provisioned.
- `backing_provider_name` - Indicates Cloud service provider on which the server for a multi-tenant cluster is provisioned.
- `provider_disk_iops` - Indicates the maximum input/output operations per second (IOPS) the system can perform. The possible values depend on the selected providerSettings.instanceSizeName and diskSizeGB.
- `provider_disk_type_name` - Describes Azure disk type of the server's root volume (Azure Only).
- `provider_encrypt_ebs_volume` - Indicates whether the Amazon EBS encryption is enabled. This feature encrypts the server's root volume for both data at rest within the volume and data moving between the volume and the instance.
- `provider_region_name` - Indicates Physical location of your MongoDB cluster. The region you choose can affect network latency for clients accessing your databases. Requires the Atlas Region name, see the reference list for AWS, GCP, Azure.
- `provider_volume_type` - Indicates the type of the volume. The possible values are: `STANDARD` and `PROVISIONED`.
- `replication_factor` - Number of replica set members. Each member keeps a copy of your databases, providing high availability and data redundancy. The possible values are 3, 5, or 7. The default value is 3.
- `replication_specs` - Configuration for cluster regions. See Replication Spec below for more details.

» BI Connector

Indicates BI Connector for Atlas configuration.

- `enabled` - Indicates whether or not BI Connector for Atlas is enabled on the cluster.

- `read_preference` - Indicates the read preference to be used by BI Connector for Atlas on the cluster. Each BI Connector for Atlas read preference contains a distinct combination of readPreference and readPreferenceTags options. For details on BI Connector for Atlas read preferences, refer to the BI Connector Read Preferences Table.

» Replication Spec

Configuration for cluster regions.

- `id` - Unique identifer of the replication document for a zone in a Global Cluster.
- `num_shards` - Number of shards to deploy in the specified zone.
- `regions_config` - Describes the physical location of the region. Each regionsConfig document describes the region's priority in elections and the number and type of MongoDB nodes Atlas deploys to the region. You must order each regionsConfigs document by regionsConfig.priority, descending. See Region Config below for more details.
- `zone_name` - Indicates the n ame for the zone in a Global Cluster.

» Region Config

Physical location of the region.

- `region_name` - Name for the region specified.
- `electable_nodes` - Number of electable nodes for Atlas to deploy to the region.
- `priority` - Election priority of the region. For regions with only read-only nodes, set this value to 0.
- `read_only_nodes` - Number of read-only nodes for Atlas to deploy to the region. Read-only nodes can never become the primary, but can facilitate local-reads. Specify 0 if you do not want any read-only nodes in the region.
- `analytics_nodes` - Indicates the number of analytics nodes for Atlas to deploy to the region. Analytics nodes are useful for handling analytic data such as reporting queries from BI Connector for Atlas. Analytics nodes are read-only, and can never become the primary.

» Labels

Contains key-value pairs that tag and categorize the cluster. Each key and value has a maximum length of 255 characters.

- `key` - The key that was set.
- `value` - The value that represents the key.

» Plugin

Contains a key-value pair that tags that the cluster was created by a Terraform Provider and notes the version.

- `name` - The name of the current plugin
- `version` - The current version of the plugin.

See detailed information for arguments and attributes: MongoDB API Clusters

» mongodbatlas_cloud_provider_snapshot

`mongodbatlas_cloud_provider_snapshot` provides an Cloud Provider Snapshot entry datasource. Atlas Cloud Provider Snapshots provide localized backup storage using the native snapshot functionality of the cluster's cloud service provider.

NOTE: Groups and projects are synonymous terms. You may find `groupId` in the official documentation.

» Example Usage

```
resource "mongodbatlas_cloud_provider_snapshot" "test" {
  group_id          = "5d0f1f73cf09a29120e173cf"
  cluster_name      = "MyClusterTest"
  description       = "SomeDescription"
  retention_in_days = 1
}

data "mongodbatlas_cloud_provider_snapshot" "test" {
  snapshot_id  = "5d1285acd5ec13b6c2d1726a"
  group_id     = "${mongodbatlas_cloud_provider_snapshot.test.group_id}"
  cluster_name = "${mongodbatlas_cloud_provider_snapshot.test.cluster_name}"
}
```

» Argument Reference

- `snapshot_id` - (Required) The unique identifier of the snapshot you want to retrieve.
- `cluster_name` - (Required) The name of the Atlas cluster that contains the snapshot you want to retrieve.
- `group_id` - (Required) The unique identifier of the project for the Atlas cluster.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - Unique identifier of the snapshot.
- `created_at` - UTC ISO 8601 formatted point in time when Atlas took the snapshot.
- `expires_at` - UTC ISO 8601 formatted point in time when Atlas will delete the snapshot.
- `description` - UDescription of the snapshot. Only present for on-demand snapshots.
- `master_key_uuid` - Unique ID of the AWS KMS Customer Master Key used to encrypt the snapshot. Only visible for clusters using Encryption at Rest via Customer KMS.
- `mongod_version` - Version of the MongoDB server.
- `snapshot_type` - Specified the type of snapshot. Valid values are onDemand and scheduled.
- `status` - Current status of the snapshot. One of the following values: queued, inProgress, completed, failed.
- `storage_size_bytes` - Specifies the size of the snapshot in bytes.
- `type` - Specifies the type of cluster: replicaSet or shardedCluster.

For more information see: MongoDB Atlas API Reference.

» mongodbatlas_cloud_provider_snapshots

`mongodbatlas_cloud_provider_snapshots` provides an Cloud Provider Snapshot entry datasource. Atlas Cloud Provider Snapshots provide localized backup storage using the native snapshot functionality of the cluster's cloud service provider.

NOTE: Groups and projects are synonymous terms. You may find `groupId` in the official documentation.

» Example Usage

```
resource "mongodbatlas_cloud_provider_snapshots" "test" {
  group_id          = "5d0f1f73cf09a29120e173cf"
  cluster_name      = "MyClusterTest"
  description       = "SomeDescription"
  retention_in_days = 1
}

data "mongodbatlas_cloud_provider_snapshots" "test" {
  group_id     = "${mongodbatlas_cloud_provider_snapshots.test.group_id}"
```

```
  cluster_name = "${mongodbatlas_cloud_provider_snapshots.test.cluster_name}"
}
```

» Argument Reference

- `cluster_name` - (Required) The name of the Atlas cluster that contains the snapshot you want to retrieve.
- `group_id` - (Required) The unique identifier of the project for the Atlas cluster.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `results` - Includes cloudProviderSnapshot object for each item detailed in the results array section.
- `totalCount` - Count of the total number of items in the result set. It may be greater than the number of objects in the results array if the entire result set is paginated.

» CloudProviderSnapshot

- `id` - Unique identifier of the snapshot.
- `created_at` - UTC ISO 8601 formatted point in time when Atlas took the snapshot.
- `expires_at` - UTC ISO 8601 formatted point in time when Atlas will delete the snapshot.
- `description` - UDescription of the snapshot. Only present for on-demand snapshots.
- `master_key_uuid` - Unique ID of the AWS KMS Customer Master Key used to encrypt the snapshot. Only visible for clusters using Encryption at Rest via Customer KMS.
- `mongod_version` - Version of the MongoDB server.
- `snapshot_type` - Specified the type of snapshot. Valid values are onDemand and scheduled.
- `status` - Current status of the snapshot. One of the following values: queued, inProgress, completed, failed.
- `storage_size_bytes` - Specifies the size of the snapshot in bytes.
- `type` - Specifies the type of cluster: replicaSet or shardedCluster.

For more information see: MongoDB Atlas API Reference.

» mongodbatlas_cloud_provider_snapshot_restore_job

`mongodbatlas_cloud_provider_snapshot_restore_job` provides a Cloud Provider Snapshot Restore Job entry datasource. Gets all cloud provider snapshot restore jobs for the specified cluster.

NOTE: Groups and projects are synonymous terms. You may find `groupId` in the official documentation.

» Example Usage

First create a snapshot of the desired cluster. Then request that snapshot be restored in an automated fashion to the designated cluster and project.

```
resource "mongodbatlas_cloud_provider_snapshot" "test" {
  project_id          = "5cf5a45a9ccf6400e60981b6"
  cluster_name      = "MyCluster"
  description       = "MyDescription"
  retention_in_days = 1
}

resource "mongodbatlas_cloud_provider_snapshot_restore_job" "test" {
  project_id     = "5cf5a45a9ccf6400e60981b6"
  cluster_name = "MyCluster"
  snapshot_id  = "${mongodbatlas_cloud_provider_snapshot.test.id}"
  delivery_type = {
    automated = true
    target_cluster_name = "MyCluster"
    target_project_id     = "5cf5a45a9ccf6400e60981b6"
  }
}

data "mongodbatlas_cloud_provider_snapshot_restore_job" "test" {
  project_id     = "${mongodbatlas_cloud_provider_snapshot_restore_job.test.project_id}"
  cluster_name = "${mongodbatlas_cloud_provider_snapshot_restore_job.test.cluster_name}"
  job_id       = "${mongodbatlas_cloud_provider_snapshot_restore_job.test.id}"
}
```

» Argument Reference

- `project_id` - (Required) The unique identifier of the project for the Atlas cluster.
- `cluster_name` - (Required) The name of the Atlas cluster for which you want to retrieve the restore job.

- `job_id` - (Required) The unique identifier of the restore job to retrieve.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `cancelled` - Indicates whether the restore job was canceled.
- `created_at` - UTC ISO 8601 formatted point in time when Atlas created the restore job.
- `delivery_type` - Type of restore job to create. Possible values are: automated and download.
- `delivery_url` - One or more URLs for the compressed snapshot files for manual download. Only visible if deliveryType is download.
- `expired` - Indicates whether the restore job expired.
- `expires_at` - UTC ISO 8601 formatted point in time when the restore job expires.
- `finished_at` - UTC ISO 8601 formatted point in time when the restore job completed.
- `id` - The unique identifier of the restore job.
- `snapshot_id` - Unique identifier of the source snapshot ID of the restore job.
- `target_group_id` - Name of the target Atlas project of the restore job. Only visible if deliveryType is automated.
- `target_cluster_name` - Name of the target Atlas cluster to which the restore job restores the snapshot. Only visible if deliveryType is automated.
- `timestamp` - Timestamp in ISO 8601 date and time format in UTC when the snapshot associated to snapshotId was taken.

For more information see: MongoDB Atlas API Reference.

» mongodbatlas_cloud_provider_snapshot_restore_jobs

`mongodbatlas_cloud_provider_snapshot_restore_jobs` provides a Cloud Provider Snapshot Restore Jobs entry datasource. Gets all cloud provider snapshot restore jobs for the specified cluster.

NOTE: Groups and projects are synonymous terms. You may find `groupId` in the official documentation.

» Example Usage

First create a snapshot of the desired cluster. Then request that snapshot be restored in an automated fashion to the designated cluster and project.

```
resource "mongodbatlas_cloud_provider_snapshot" "test" {
  project_id          = "5cf5a45a9ccf6400e60981b6"
  cluster_name      = "MyCluster"
  description       = "MyDescription"
  retention_in_days = 1
}

resource "mongodbatlas_cloud_provider_snapshot_restore_job" "test" {
  project_id     = "5cf5a45a9ccf6400e60981b6"
  cluster_name = "MyCluster"
  snapshot_id  = "${mongodbatlas_cloud_provider_snapshot.test.id}"
  delivery_type = {
    automated = true
    target_cluster_name = "MyCluster"
    target_project_id     = "5cf5a45a9ccf6400e60981b6"
  }
}

data "mongodbatlas_cloud_provider_snapshot_restore_jobs" "test" {
  project_id     = "${mongodbatlas_cloud_provider_snapshot_restore_job.test.project_id}"
  cluster_name = "${mongodbatlas_cloud_provider_snapshot_restore_job.test.cluster_name}"
}
```

» Argument Reference

- `project_id` - (Required) The unique identifier of the project for the Atlas cluster.
- `cluster_name` - (Required) The name of the Atlas cluster for which you want to retrieve restore jobs.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `results` - Includes cloudProviderSnapshotRestoreJob object for each item detailed in the results array section.
- `totalCount` - Count of the total number of items in the result set. It may be greater than the number of objects in the results array if the entire result set is paginated.

» CloudProviderSnapshotRestoreJob

- `cancelled` - Indicates whether the restore job was canceled.

- `created_at` - UTC ISO 8601 formatted point in time when Atlas created the restore job.
- `delivery_type` - Type of restore job to create. Possible values are: automated and download.
- `delivery_url` - One or more URLs for the compressed snapshot files for manual download. Only visible if deliveryType is download.
- `expired` - Indicates whether the restore job expired.
- `expires_at` - UTC ISO 8601 formatted point in time when the restore job expires.
- `finished_at` - UTC ISO 8601 formatted point in time when the restore job completed.
- `id` - The unique identifier of the restore job.
- `snapshot_id` - Unique identifier of the source snapshot ID of the restore job.
- `target_group_id` - Name of the target Atlas project of the restore job. Only visible if deliveryType is automated.
- `target_cluster_name` - Name of the target Atlas cluster to which the restore job restores the snapshot. Only visible if deliveryType is automated.
- `timestamp` - Timestamp in ISO 8601 date and time format in UTC when the snapshot associated to snapshotId was taken.

For more information see: MongoDB Atlas API Reference.

» mongodbatlas_network_container

`mongodbatlas_network_container` describes a Network Peering Container. The resource requires your Project ID and container ID.

IMPORTANT: This resource creates one Network Peering container into which Atlas can deploy Network Peering connections. An Atlas project can have a maximum of one container for each cloud provider. You must have either the Project Owner or Organization Owner role to successfully call this endpoint.

NOTE: Groups and projects are synonymous terms. You may find **group_id** in the official documentation.

» Example Usage

» Basic Example.

```
resource "mongodbatlas_network_container" "test" {
  project_id       = "<YOUR-PROJECT-ID>"
  atlas_cidr_block = "10.8.0.0/21"
  provider_name    = "AWS"
```

```
  region_name      = "US_EAST_1"
}

data "mongodbatlas_network_container" "test" {
    project_id          = mongodbatlas_network_container.test.project_id
    container_id        = mongodbatlas_network_container.test.id
}
```

» Argument Reference

- `project_id` - (Required) The unique ID for the project to create the database user.
- `container_id` - (Required) The Network Peering Container ID.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The Network Peering Container ID.
- `atlas_cidr_block` - CIDR block that Atlas uses for your clusters. Atlas uses the specified CIDR block for all other Network Peering connections created in the project. The Atlas CIDR block must be at least a /24 and at most a /21 in one of the following private networks.
- `provider_name` - Cloud provider for this Network Peering connection. If omitted, Atlas sets this parameter to AWS.
- `region_name` - AWS region.
- `region` - Azure region where the container resides.
- `azure_subscription_id` - Unique identifer of the Azure subscription in which the VNet resides.
- `provisioned` - Indicates whether the project has Network Peering connections deployed in the container.
- `gcp_project_id` - Unique identifier of the GCP project in which the Network Peering connection resides.
- `network_name` - Name of the Network Peering connection in the Atlas project.
- `vpc_id` - Unique identifier of the project's VPC.
- `vnet_name` - The name of the Azure VNet. This value is null until you provision an Azure VNet in the container.

See detailed information for arguments and attributes: MongoDB API Network Peering Container

» mongodbatlas_network_containers

`mongodbatlas_network_containers` describes all Network Peering Containers. The data source requires your Project ID.

NOTE: Groups and projects are synonymous terms. You may find **group_id** in the official documentation.

» Example Usage

» Basic Example.

```
resource "mongodbatlas_network_container" "test" {
  project_id       = "<YOUR-PROJECT-ID>"
  atlas_cidr_block = "10.8.0.0/21"
  provider_name    = "AWS"
  region_name      = "US_EAST_1"
}

data "mongodbatlas_network_containers" "test" {
    project_id     = mongodbatlas_network_container.test.project_id
  provider_name  = mongodbatlas_network_container.test.provider_name //"AWS"
}
```

» Argument Reference

- `project_id` - (Required) The unique ID for the project to create the database user.
- `provider_name` - (Required) Cloud provider for this Network peering container. Accepted values are AWS, GCP, and Azure.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - Autogenerated Unique ID for this data source.
- `results` - A list where each represents a Network Peering Container.

» Network Peering Container

- `id` - The Network Peering Container ID.

- `atlas_cidr_block` - CIDR block that Atlas uses for your clusters. Atlas uses the specified CIDR block for all other Network Peering connections created in the project. The Atlas CIDR block must be at least a /24 and at most a /21 in one of the following private networks.
- `provider_name` - Cloud provider for this Network Peering connection. If omitted, Atlas sets this parameter to AWS.
- `region_name` - AWS region.
- `region` - Azure region where the container resides.
- `azure_subscription_id` - Unique identifer of the Azure subscription in which the VNet resides.
- `provisioned` - Indicates whether the project has Network Peering connections deployed in the container.
- `gcp_project_id` - Unique identifier of the GCP project in which the Network Peering connection resides.
- `network_name` - Name of the Network Peering connection in the Atlas project.
- `vpc_id` - Unique identifier of the project's VPC.
- `vnet_name` - The name of the Azure VNet. This value is null until you provision an Azure VNet in the container.

See detailed information for arguments and attributes: MongoDB API Network Peering Container

» mongodbatlas_network_peering

`mongodbatlas_network_peering` describes a Network Peering Connection.

NOTE: Groups and projects are synonymous terms. You may find **group_id** in the official documentation.

» Example Usage

» Basic Example (AWS).

```
resource "mongodbatlas_network_peering" "test" {
    accepter_region_name    = "us-east-1"
    project_id              = "<YOUR-PROJEC-ID>"
    container_id            = "507f1f77bcf86cd799439011"
    provider_name           = "AWS"
    route_table_cidr_block  = "192.168.0.0/24"
    vpc_id                  = "vpc-abc123abc123"
    aws_account_id          = "abc123abc123"
}
```

```
data "mongodbatlas_network_peering" "test" {
    project_id = mongodbatlas_network_peering.test.project_id
    peering_id = mongodbatlas_network_peering.test.id
}
```

» Argument Reference

- `project_id` - (Required) The unique ID for the project to create the database user.
- `peering_id` - (Required) Atlas assigned unique ID for the peering connection.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The Network Peering Connection ID.
- `connection_id` - Unique identifier for the peering connection.
- `accepter_region_name` - Specifies the region where the peer VPC resides. For complete lists of supported regions, see Amazon Web Services.
- `aws_account_id` - Account ID of the owner of the peer VPC.
- `provider_name` - Cloud provider for this VPC peering connection. If omitted, Atlas sets this parameter to AWS. (Possible Values `AWS`, `AZURE`, `GCP`).
- `route_table_cidr_block` - Peer VPC CIDR block or subnet.
- `vpc_id` - Unique identifier of the peer VPC.
- `error_state_name` - Error state, if any. The VPC peering connection error state value can be one of the following: `REJECTED`, `EXPIRED`, `INVALID_ARGUMENT`.
- `status_name` - The VPC peering connection status value can be one of the following: `INITIATING`, `PENDING_ACCEPTANCE`, `FAILED`, `FINALIZING`, `AVAILABLE`, `TERMINATING`.
- `atlas_cidr_block` - Unique identifier for an Azure AD directory.
- `azure_directory_id` - Unique identifier for an Azure AD directory.
- `azure_subscription_id` - Unique identifer of the Azure subscription in which the VNet resides.
- `resource_group_name` - Name of your Azure resource group.
- `vnet_name` - Name of your Azure VNet.
- `error_state` - Description of the Atlas error when `status` is `Failed`, Otherwise, Atlas returns `null`.
- `status` - Status of the Atlas network peering connection: `ADDING_PEER`, `AVAILABLE`, `FAILED`, `DELETING`, `WAITING_FOR_USER`.
- `gcp_project_id` - GCP project ID of the owner of the network peer.

- `network_name` - Name of the network peer to which Atlas connects.
- `error_message` - When `"status" : "FAILED"`, Atlas provides a description of the error.

See detailed information for arguments and attributes: MongoDB API Network Peering Connection

» mongodbatlas_network_peering

`mongodbatlas_network_peerings` describes all Network Peering Connections.

NOTE: Groups and projects are synonymous terms. You may find **group_id** in the official documentation.

» Example Usage

» Basic Example (AWS).

```
resource "mongodbatlas_network_peering" "test" {
    accepter_region_name    = "us-east-1"
    project_id              = "<YOUR-PROJEC-ID>"
    container_id            = "507f1f77bcf86cd799439011"
    provider_name           = "AWS"
    route_table_cidr_block  = "192.168.0.0/24"
    vpc_id                  = "vpc-abc123abc123"
    aws_account_id          = "abc123abc123"
}


data "mongodbatlas_network_peerings" "test" {
    project_id = mongodbatlas_network_peering.test.project_id
}
```

» Argument Reference

- `project_id` - (Required) The unique ID for the project to create the database user.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The Network Peering Connection ID.

- `results` - A list where each represents a Network Peering Connection.

» Network Peering Connection

- `peering_id` - Atlas assigned unique ID for the peering connection.
- `connection_id` - Unique identifier for the peering connection.
- `accepter_region_name` - Specifies the region where the peer VPC resides. For complete lists of supported regions, see Amazon Web Services.
- `aws_account_id` - Account ID of the owner of the peer VPC.
- `provider_name` - Cloud provider for this VPC peering connection. If omitted, Atlas sets this parameter to AWS. (Possible Values `AWS`, `AZURE`, `GCP`).
- `route_table_cidr_block` - Peer VPC CIDR block or subnet.
- `vpc_id` - Unique identifier of the peer VPC.
- `error_state_name` - Error state, if any. The VPC peering connection error state value can be one of the following: `REJECTED`, `EXPIRED`, `INVALID_ARGUMENT`.
- `status_name` - The VPC peering connection status value can be one of the following: `INITIATING`, `PENDING_ACCEPTANCE`, `FAILED`, `FINALIZING`, `AVAILABLE`, `TERMINATING`.
- `atlas_cidr_block` - Unique identifier for an Azure AD directory.
- `azure_directory_id` - Unique identifier for an Azure AD directory.
- `azure_subscription_id` - Unique identifer of the Azure subscription in which the VNet resides.
- `resource_group_name` - Name of your Azure resource group.
- `vnet_name` - Name of your Azure VNet.
- `error_state` - Description of the Atlas error when `status` is `Failed`, Otherwise, Atlas returns `null`.
- `status` - Status of the Atlas network peering connection: `ADDING_PEER`, `AVAILABLE`, `FAILED`, `DELETING`, `WAITING_FOR_USER`.
- `gcp_project_id` - GCP project ID of the owner of the network peer.
- `network_name` - Name of the network peer to which Atlas connects.
- `error_message` - When `"status" : "FAILED"`, Atlas provides a description of the error.

See detailed information for arguments and attributes: MongoDB API Network Peering Connection

» mongodbatlas_maintenance_window

`mongodbatlas_maintenance_window` provides a Maintenance Window entry datasource. Gets information regarding the configured maintenance window for a MongoDB Atlas project.

NOTE: Groups and projects are synonymous terms. You may find `groupId` in the official documentation.

» Examples Usage

```
resource "mongodbatlas_maintenance_window" "test" {
  project_id  = "<your-project-id>"
  day_of_week = 3
  hour_of_day = 4
}

data "mongodbatlas_maintenance_window" "test" {
  project_id = "${mongodbatlas_maintenance_window.test.id}"
}
```

```
resource "mongodbatlas_maintenance_window" "test" {
  project_id  = "<your-project-id>"
  start_asap  = true
}

data "mongodbatlas_maintenance_window" "test" {
  project_id = "${mongodbatlas_maintenance_window.test.id}"
}
```

» Argument Reference

- `project_id` - The unique identifier of the project for the Maintenance Window.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `day_of_week` - Day of the week when you would like the maintenance window to start as a 1-based integer: S=1, M=2, T=3, W=4, T=5, F=6, S=7.
- `hour_of_day` - Hour of the day when you would like the maintenance window to start. This parameter uses the 24-hour clock, where midnight is 0, noon is 12 (Time zone is UTC).
- `start_asap` - Flag indicating whether project maintenance has been directed to start immediately. If you request that maintenance begin immediately, this field returns true from the time the request was made until the time the maintenance event completes.

- `number_of_deferrals` - Number of times the current maintenance event for this project has been deferred, you can set a maximum of 2 deferrals.

For more information see: MongoDB Atlas API Reference.

» mongodbatlas_auditing

`mongodbatlas_auditing` describes a Auditing.

NOTE: Groups and projects are synonymous terms. You may find **group_id** in the official documentation.

» Example Usage

```
resource "mongodbatlas_auditing" "test" {
    project_id                 = "<project-id>"
    audit_filter               = "{ 'atype': 'authenticate', 'param': {   'user': 'auditAdm
    audit_authorization_success = false
    enabled                    = true
}

data "mongodbatlas_auditing" "test" {
            project_id = "${mongodbatlas_auditing.test.id}"
        }
```

» Argument Reference

- `project_id` - (Required) The unique ID for the project to create the database user.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `configuration_type` - Denotes the configuration method for the audit filter. Possible values are: NONE - auditing not configured for the project.m FILTER_BUILDER - auditing configured via Atlas UI filter builderm FILTER_JSON - auditing configured via Atlas custom filter or API.
- `audit_filter` - Indicates whether the auditing system captures successful authentication attempts for audit filters using the "atype" : "authCheck" auditing event. For more information, see auditAuthorizationSuccess
- `audit_authorization_success` - JSON-formatted audit filter used by the project

- `enabled` - Denotes whether or not the project associated with the {GROUP-ID} has database auditing enabled.

See detailed information for arguments and attributes: MongoDB API Auditing

» mongodbatlas_custom_db_roles

`mongodbatlas_custom_db_roles` describe all Custom DB Roles. This represents a custom db roles.

NOTE: Groups and projects are synonymous terms. You may find `groupId` in the official documentation.

» Example Usage

```
resource "mongodbatlas_database_user" "test" {
  username      = "test-acc-username"
  password      = "test-acc-password"
  project_id    = "<PROJECT-ID>"
  database_name = "admin"

  roles {
    role_name     = "readWrite"
    database_name = "admin"
  }

  roles {
    role_name     = "atlasAdmin"
    database_name = "admin"
  }
}

data "mongodbatlas_custom_db_roles" "test" {
  project_id = "${mongodbatlas_custom_db_role.test.project_id}"
}
```

» Argument Reference

- `project_id` - (Required) The unique ID for the project to get all custom db roles.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - Autogenerated Unique ID for this data source.
- `results` - A list where each represents a custom db roles.

» Actions

Each object in the actions array represents an individual privilege action granted by the role. It is an required field.

- `action` - (Required) Name of the privilege action. For a complete list of actions available in the Atlas API, see Custom Role Actions.
- `resources` - (Required) Contains information on where the action is granted. Each object in the array either indicates a database and collection on which the action is granted, or indicates that the action is granted on the cluster resource.
- `resources.#.collection` - (Optional) Collection on which the action is granted. If this value is an empty string, the action is granted on all collections within the database specified in the actions.resources.db field.
- `resources.#.database_name` Database on which the action is granted.
- `resources.#.cluster` (Optional) Set to true to indicate that the action is granted on the cluster resource.

» Inherited Roles

Each object in the inheritedRoles array represents a key-value pair indicating the inherited role and the database on which the role is granted. It is an optional field.

- `database_name` (Required) Database on which the inherited role is granted.
- `role_name` (Required) Name of the inherited role. This can either be another custom role or a built-in role.

See MongoDB Atlas API Documentation for more information.

» mongodbatlas_alert_configuration

`mongodbatlas_alert_configuration` describes an Alert Configuration.

NOTE: Groups and projects are synonymous terms. You may find **group_id** in the official documentation.

» Example Usage

```
resource "mongodbatlas_alert_configuration" "test" {
  project_id = "<PROJECT-ID>"
  event_type = "OUTSIDE_METRIC_THRESHOLD"
  enabled    = true

  notification {
    type_name     = "GROUP"
    interval_min  = 5
    delay_min     = 0
    sms_enabled   = false
    email_enabled = true
  }

  matcher {
    field_name = "HOSTNAME_AND_PORT"
    operator   = "EQUALS"
    value      = "SECONDARY"
  }

  metric_threshold = {
    metric_name = "ASSERT_REGULAR"
    operator    = "LESS_THAN"
    threshold   = 99.0
    units       = "RAW"
    mode        = "AVERAGE"
  }
}


data "mongodbatlas_alert_configuration" "test" {
    project_id             = "${mongodbatlas_alert_configuration.test.project_id}"
    alert_configuration_id = "${mongodbatlas_alert_configuration.test.alert_configuration_id
}
```

» Argument Reference

- `project_id` - (Required) The ID of the project where the alert configuration will create.

- `alert_configuration_id` - (Required) Unique identifier for the alert configuration.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `group_id` - Unique identifier of the project that owns this alert configuration.
- `created` - Timestamp in ISO 8601 date and time format in UTC when this alert configuration was created.
- `updated` - Timestamp in ISO 8601 date and time format in UTC when this alert configuration was last updated.
- `enabled` - If set to true, the alert configuration is enabled. If enabled is not exported it is set to false.
- `event_type` - The type of event that will trigger an alert. Alert type. Possible values:
  - Host
    * `OUTSIDE_METRIC_THRESHOLD`
    * `HOST_RESTARTED`
    * `HOST_UPGRADED`
    * `HOST_NOW_SECONDARY`
    * `HOST_NOW_PRIMARY`
    * Replica set
    * `NO_PRIMARY`
    * `TOO_MANY_ELECTIONS` Sharded cluster
    * `CLUSTER_MONGOS_IS_MISSING`
    * `User`
    * `JOINED_GROUP`
    * `REMOVED_FROM_GROUP`
    * `USER_ROLES_CHANGED_AUDIT`
    * Project
    * `USERS_AWAITING_APPROVAL`
    * `USERS_WITHOUT_MULTI_FACTOR_AUTH`
    * `GROUP_CREATED`
    * Team
    * `JOINED_TEAM`

* REMOVED_FROM_TEAM
* Organization

* INVITED_TO_ORG
* JOINED_ORG
* Data Explorer

* DATA_EXPLORER
* DATA_EXPLORER_CRUD
* Billing

* CREDIT_CARD_ABOUT_TO_EXPIRE
* CHARGE_SUCCEEDED
* INVOICE_CLOSED

NOTE: If this is set to OUTSIDE_METRIC_THRESHOLD, the metricThreshold field must also be set.

» Matchers

Rules to apply when matching an object against this alert configuration. Only entities that match all these rules are checked for an alert condition. You can filter using the matchers array only when the eventTypeName specifies an event for a host, replica set, or sharded cluster.

- `field_name` - Name of the field in the target object to match on. Host alerts support these fields:
  - TYPE_NAME
  - HOSTNAME
  - PORT
  - HOSTNAME_AND_PORT
  - REPLICA_SET_NAME Replica set alerts support these fields:
  - REPLICA_SET_NAME
  - SHARD_NAME
  - CLUSTER_NAME Sharded cluster alerts support these fields:
  - CLUSTER_NAME
  - SHARD_NAME

All other types of alerts do not support matchers.

- `operator` - If omitted, the configuration is disabled.
- `value` - If omitted, the configuration is disabled.
- `operator` - The operator to test the field's value. Accepted values are:
  - EQUALS
  - NOT_EQUALS

  - CONTAINS
  - NOT_CONTAINS
  - STARTS_WITH
  - ENDS_WITH
  - REGEX

- `value` - Value to test with the specified operator. If `field_name` is set to TYPE_NAME, you can match on the following values:
  - PRIMARY
  - SECONDARY
  - STANDALONE
  - CONFIG
  - MONGOS

» Metric Threshold

The threshold that causes an alert to be triggered. Required if `event_type_name` : "OUTSIDE_METRIC_THRESHOLD".

- `metric_name` - Name of the metric to check.
- `operator` - Operator to apply when checking the current metric value against the threshold value. Accepted values are:
  - GREATER_THAN
  - LESS_THAN
- `threshold` - Threshold value outside of which an alert will be triggered.
- `units` - The units for the threshold value. Depends on the type of metric. Accepted values are:
  - RAW
  - BITS
  - BYTES
  - KILOBITS
  - KILOBYTES
  - MEGABITS
  - MEGABYTES
  - GIGABITS
  - GIGABYTES
  - TERABYTES
  - PETABYTES
  - MILLISECONDS
  - SECONDS
  - MINUTES
  - HOURS
  - DAYS

- `mode` - This must be set to AVERAGE. Atlas computes the current metric value as an average.

» Notifications

Notifications to send when an alert condition is detected.

- `api_token` - Slack API token. Required for the SLACK notifications type. If the token later becomes invalid, Atlas sends an email to the project owner and eventually removes the token.
- `channel_name` - Slack channel name. Required for the SLACK notifications type.
- `datadog_api_key` - Datadog API Key. Found in the Datadog dashboard. Required for the DATADOG notifications type.
- `datadog_region` - Region that indicates which API URL to use. Accepted regions are: `US`, `EU`. The default Datadog region is US.
- `delay_min` - Number of minutes to wait after an alert condition is detected before sending out the first notification.
- `email_address` - Email address to which alert notifications are sent. Required for the EMAIL notifications type.
- `email_enabled` - Flag indicating if email notifications should be sent. Configurable for `ORG`, `GROUP`, and `USER` notifications types.
- `flowdock_api_token` - The Flowdock personal API token. Required for the `FLOWDOCK` notifications type. If the token later becomes invalid, Atlas sends an email to the project owner and eventually removes the token.
- `flow_name` - Flowdock flow name in lower-case letters. Required for the `FLOWDOCK` notifications type
- `interval_min` - Number of minutes to wait between successive notifications for unacknowledged alerts that are not resolved. The minimum value is 5.
- `mobile_number` - Mobile number to which alert notifications are sent. Required for the SMS notifications type.
- `ops_genie_api_key` - Opsgenie API Key. Required for the `OPS_GENIE` notifications type. If the key later becomes invalid, Atlas sends an email to the project owner and eventually removes the token.
- `ops_genie_region` - Region that indicates which API URL to use. Accepted regions are: `US` ,`EU`. The default Opsgenie region is US.

- `org_name` - Flowdock organization name in lower-case letters. This is the name that appears after www.flowdock.com/app/ in the URL string. Required for the FLOWDOCK notifications type.
- `service_key` - PagerDuty service key. Required for the PAGER_DUTY notifications type. If the key later becomes invalid, Atlas sends an email to the project owner and eventually removes the key.
- `sms_enabled` - Flag indicating if text message notifications should be sent. Configurable for `ORG`, `GROUP`, and `USER` notifications types.
- `team_id` - Unique identifier of a team.
- `type_name` - Type of alert notification. Accepted values are:
  - `DATADOG`
  - `EMAIL`
  - `FLOWDOCK`
  - `GROUP` (Project)
  - `OPS_GENIE`
  - `ORG`
  - `PAGER_DUTY`
  - `SLACK`
  - `SMS`
  - `TEAM`
  - `USER`
  - `VICTOR_OPS`
  - `WEBHOOK`
- `username` - Name of the Atlas user to which to send notifications. Only a user in the project that owns the alert configuration is allowed here. Required for the `USER` notifications type.
- `victor_ops_api_key` - VictorOps API key. Required for the `VICTOR_OPS` notifications type. If the key later becomes invalid, Atlas sends an email to the project owner and eventually removes the key.
- `victor_ops_routing_key` - VictorOps routing key. Optional for the `VICTOR_OPS` notifications type. If the key later becomes invalid, Atlas sends an email to the project owner and eventually removes the key.

See detailed information for arguments and attributes: MongoDB API Alert Configuration

» mongodbatlas_teams

`mongodbatlas_teams` describes a Team. The resource requires your Organization ID, Project ID and Team ID.

NOTE: Groups and projects are synonymous terms. You may find group_id in the official documentation.

» Example Usage

```
resource "mongodbatlas_teams" "test" {
  org_id     = "<ORGANIZATION-ID>"
  name       = "myNewTeam"
  usernames  = ["user1", "user2", "user3"]
}

data "mongodbatlas_teams" "test" {
    org_id     = mongodbatlas_teams.test.org_id
    team_id    = mongodbatlas_teams.test.team_id
}
```

» Argument Reference

- `org_id` - (Required) The unique identifier for the organization you want to associate the team with.
- `team_id` - (Required) The unique identifier for the team.

» Attributes Reference

In addition to all arguments above, the following attributes are exported: * `id` - The Terraform's unique identifier used internally for state management. * `name` - The name of the team you want to create. * `usernames` - The users who are part of the organization.

See detailed information for arguments and attributes: MongoDB API Teams

» mongodbatlas_private_endpoint

`mongodbatlas_private_endpoint` describe a Private Endpoint. This represents a Private Endpoint Connection to retrieve details regarding a private endpoint by id in an Atlas project

NOTE: Groups and projects are synonymous terms. You may find group_id in the official documentation.

» Example Usage

```
resource "mongodbatlas_private_endpoint" "test" {
  project_id    = "<PROJECT-ID>"
  provider_name = "AWS"
  region        = "us-east-1"
}

data "mongodbatlas_private_endpoint" "test" {
    project_id      = "${mongodbatlas_private_endpoint.test.project_id}"
    private_link_id = "${mongodbatlas_private_endpoint.test.private_link_id}"
}
```

» Argument Reference

- `project_id` - (Required) Unique identifier for the project.
- `private_link_id` - (Required) Unique identifier of the AWS PrivateLink connection.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The Terraform's unique identifier used internally for state management.
- `endpoint_service_name` - Name of the PrivateLink endpoint service in AWS. Returns null while the endpoint service is being created.
- `error_message` - Error message pertaining to the AWS PrivateLink connection. Returns null if there are no errors.
- `interface_endpoints` - Unique identifiers of the interface endpoints in your VPC that you added to the AWS PrivateLink connection.
- `status` - Status of the AWS PrivateLink connection. Returns one of the following values:
  - `INITIATING` Atlas is creating the network load balancer and VPC endpoint service.
  - `WAITING_FOR_USER` The Atlas network load balancer and VPC endpoint service are created and ready to receive connection requests. When you receive this status, create an interface endpoint to continue configuring the AWS PrivateLink connection.
  - `FAILED` A system failure has occurred.
  - `DELETING` The AWS PrivateLink connection is being deleted.

See MongoDB Atlas API Documentation for more information.

» mongodbatlas_private_endpoint_link

`mongodbatlas_private_endpoint_link` describe a Private Endpoint Link. This represents a Private Endpoint Link Connection that wants to retrieve details in an Atlas project.

NOTE: Groups and projects are synonymous terms. You may find group_id in the official documentation.

» Example Usage

```
resource "mongodbatlas_private_endpoint" "test" {
  project_id    = "<PROJECT_ID>"
  provider_name = "AWS"
  region        = "us-east-1"
}

resource "aws_vpc_endpoint" "ptfe_service" {
  vpc_id             = "vpc-7fc0a543"
  service_name       = "${mongodbatlas_private_endpoint.test.endpoint_service_name}"
  vpc_endpoint_type  = "Interface"
  subnet_ids         = ["subnet-de0406d2"]
  security_group_ids = ["sg-3f238186"]
}

resource "mongodbatlas_private_endpoint_link" "test" {
  project_id            = "${mongodbatlas_private_endpoint.test.project_id}"
  private_link_id       = "${mongodbatlas_private_endpoint.test.private_link_id}"
  interface_endpoint_id = "${aws_vpc_endpoint.ptfe_service.id}"
}

data "mongodbatlas_private_endpoint_link" "test" {
  project_id            = "${mongodbatlas_private_endpoint_link.test.project_id}"
  private_link_id       = "${mongodbatlas_private_endpoint_link.test.private_link_id}"
  interface_endpoint_id = "${mongodbatlas_private_endpoint_link.test.interface_endpoint_id}'
}
```

» Argument Reference

- `project_id` - (Required) Unique identifier for the project.
- `private_link_id` - (Required) Unique identifier of the AWS PrivateLink connection.
- `interface_endpoints` - (Required) Unique identifiers of the interface endpoints in your VPC.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The Terraform's unique identifier used internally for state management.
- `delete_requested` - Indicates if Atlas received a request to remove the interface endpoint from the private endpoint connection.
- `error_message` - Error message pertaining to the interface endpoint. Returns null if there are no errors.
- `connection_status` - Status of the interface endpoint. Returns one of the following values:
  - `NONE` - Atlas created the network load balancer and VPC endpoint service, but AWS hasn't yet created the VPC endpoint.
  - `PENDING_ACCEPTANCE` - AWS has received the connection request from your VPC endpoint to the Atlas VPC endpoint service.
  - `PENDING` - AWS is establishing the connection between your VPC endpoint and the Atlas VPC endpoint service.
  - `AVAILABLE` - Atlas VPC resources are connected to the VPC endpoint in your VPC. You can connect to Atlas clusters in this region using AWS PrivateLink.
  - `REJECTED` - AWS failed to establish a connection between Atlas VPC resources to the VPC endpoint in your VPC.
  - `DELETING` - Atlas is removing the interface endpoint from the private endpoint connection.

See MongoDB Atlas API Documentation for more information.

» mongodbatlas_x509_authentication_database_user

`mongodbatlas_x509_authentication_database_user` describe a X509 Authentication Database User. This represents a X509 Authentication Database User.

NOTE: Groups and projects are synonymous terms. You may find group_id in the official documentation.

» Example Usages

» Example Usage: Generate an Atlas-managed X.509 certificate for a MongoDB user

```
resource "mongodbatlas_database_user" "user" {
  project_id    = "<PROJECT-ID>"
  username      = "myUsername"
```

```
  x509_type     = "MANAGED"
  database_name = "$external"

  roles {
    role_name     = "atlasAdmin"
    database_name = "admin"
  }

  labels {
    key   = "My Key"
    value = "My Value"
  }
}

resource "mongodbatlas_x509_authentication_database_user" "test" {
  project_id              = "${mongodbatlas_database_user.user.project_id}"
  username                = "${mongodbatlas_database_user.user.username}"
  months_until_expiration = 2
}

data "mongodbatlas_x509_authentication_database_user" "test" {
  project_id = "${mongodbatlas_x509_authentication_database_user.test.project_id}"
  username   = "${mongodbatlas_x509_authentication_database_user.test.username}"
}
```

» Example Usage: Save a customer-managed X.509 configuration for an Atlas project

```
resource "mongodbatlas_x509_authentication_database_user" "test" {
  project_id        = "<PROJECT-ID>"
  customer_x509_cas = <<-EOT
    -----BEGIN CERTIFICATE-----
    MIICmTCCAgICCQDZnHzklxsT9TANBgkqhkiG9w0BAQsFADCBkDELMAkGA1UEBhMC
    VVMxDjAMBgNVBAgMBVRleGFzMQ8wDQYDVQQHDAZBdXN0aW4xETAPBgNVBAoMCHRl
    c3QuY29tMQ0wCwYDVQQLDARUZXN0MREwDwYDVQQDDAh0ZXN0LmNvbTErMCkGCSqG
    SIb3DQEJARYcbWVsaXNzYS5wbHVua2V0dEBtb25nb2RiLmNvbTAeFw0yMDAyMDQy
    MDQ2MDFaFw0yMTAyMDMyMDQ2MDFaMIGQMQswCQYDVQQGEwJVUzEOMAwGA1UECAwF
    VGV4YXMxDzANBgNVBAcMBkF1c3RpbjERMA8GA1UECgwIdGVzdC5jb20xDTALBgNV
    BAsMBFRlc3QxETAPBgNVBAMMCHRlc3QuY29tMSswKQYJKoZIhvcNAQkBFhxtZWxp
    c3NhLnBsdW5rZXR0QG1vbmdvZGIuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCB
    iQKBgQCf1LRqr1zftzdYx2Aj9G76tb0noMPtj6faGLlPji1+m6Rn7RWD9L0ntWAr
    cURxvypa9jZ9MXFzDtLevvd3tHEmfrUT3ukNDX6+Jtc4kWm+Dh2A70Pd+deKZ2/O
    Fh8audEKAESGXnTbeJCeQa1XKlIkjqQHBNwES5h1b9vJtFoLJwIDAQABMA0GCSqG
    SIb3DQEBCwUAA4GBADMUncjEPV/MiZUcVNGmktP6BPmEqMXQWUDpdGW2+Tg2JtUA
    7MMILtepBkFzLO+GlpZxeAlXO0wxiNgEmCRONgh4+t2w3e7a8GFijYQ99FHrAC5A
```

```
    iul59bdl18gVqXia1Yeq/iK7Ohfy/Jwd7Hsm530elwkM/ZEkYDjBlZSXYdyz
    -----END CERTIFICATE-----"
  EOT
}

data "mongodbatlas_x509_authentication_database_user" "test" {
  project_id = "${mongodbatlas_x509_authentication_database_user.test.project_id}"
}
```

» Argument Reference

- `project_id` - (Required) Identifier for the Atlas project associated with the X.509 configuration.
- `username` - (Optional) Username of the database user to create a certificate for.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `current_certificate` - Contains the last X.509 certificate and private key created for a database user.

Certificates * `certificates` - Array of objects where each details one unexpired database user certificate.

- `certificates.#.id` - Serial number of this certificate.
- `certificates.#.created_at` - Timestamp in ISO 8601 date and time format in UTC when Atlas created this X.509 certificate.
- `certificates.#.group_id` - Unique identifier of the Atlas project to which this certificate belongs.
- `certificates.#.not_after` - Timestamp in ISO 8601 date and time format in UTC when this certificate expires.
- `certificates.#.subject` - Fully distinguished name of the database user to which this certificate belongs. To learn more, see RFC 2253.

See MongoDB Atlas - X509 User Certificates and MongoDB Atlas - Current X509 Configuratuion Documentation for more information.

» mongodbatlas_database_user

`mongodbatlas_database_user` provides a Database User resource. This represents a database user which will be applied to all clusters within the project.

Each user has a set of roles that provide access to the project's databases. User's roles apply to all the clusters in the project: if two clusters have a `products` database and a user has a role granting `read` access on the products database, the user has that access on both clusters.

NOTE: Groups and projects are synonymous terms. You may find group_id in the official documentation.

IMPORTANT: All arguments including the password will be stored in the raw state as plain-text. Read more about sensitive data in state.

» Example Usages

```
resource "mongodbatlas_database_user" "test" {
  username           = "test-acc-username"
  password           = "test-acc-password"
  project_id         = "<PROJECT-ID>"
  auth_database_name = "admin"

  roles {
    role_name     = "readWrite"
    database_name = "dbforApp"
  }

  roles {
    role_name     = "readAnyDatabase"
    database_name = "admin"
  }

  labels {
    key   = "My Key"
    value = "My Value"
  }
}

resource "mongodbatlas_database_user" "test" {
  username           = "test-acc-username"
  x509_type          = "MANAGED"
  project_id         = "<PROJECT-ID>"
  auth_database_name = "$external"

  roles {
    role_name     = "readAnyDatabase"
    database_name = "admin"
  }
```

```
  labels {
    key   = "%s"
    value = "%s"
  }
}
```

» Argument Reference

- `auth_database_name` - (Required) The user's authentication database. A user must provide both a username and authentication database to log into MongoDB. In Atlas deployments of MongoDB, the authentication database is always the admin database.
- `project_id` - (Required) The unique ID for the project to create the database user.
- `roles` - (Required) List of user's roles and the databases / collections on which the roles apply. A role allows the user to perform particular actions on the specified database. A role on the admin database can include privileges that apply to the other databases as well. See Roles below for more details.
- `username` - (Required) Username for authenticating to MongoDB.
- `password` - (Required) User's initial password. A value is required to create the database user, however the argument but may be removed from your Terraform configuration after user creation without impacting the user, password or Terraform management. IMPORTANT --- Passwords may show up in Terraform related logs and it will be stored in the Terraform state file as plain-text. Password can be changed after creation using your preferred method, e.g. via the MongoDB Atlas UI, to ensure security. If you do change management of the password to outside of Terraform be sure to remove the argument from the Terraform configuration so it is not inadvertently updated to the original password.
- `x509_type` - (Optional) X.509 method by which the provided username is authenticated. If no value is given, Atlas uses the default value of NONE. The accepted types are:
  - `NONE` - The user does not use X.509 authentication.
  - `MANAGED` - The user is being created for use with Atlas-managed X.509.Externally authenticated users can only be created on the `$external` database.
  - `CUSTOMER` - The user is being created for use with Self-Managed X.509. Users created with this x509Type require a Common Name (CN) in the username field. Externally authenticated users can only be created on the `$external` database.

» Roles

Block mapping a user's role to a database / collection. A role allows the user to perform particular actions on the specified database. A role on the admin database can include privileges that apply to the other databases as well.

NOTE: The available privilege actions for custom MongoDB roles support a subset of MongoDB commands. See Unsupported Commands in M10+ Clusters for more information.

IMPORTANT: If a user is assigned a custom MongoDB role, they cannot be assigned any other roles.

- `name` - (Required) Name of the role to grant. See Create a Database User `roles.roleName` for valid values and restrictions.
- `database_name` - (Required) Database on which the user has the specified role. A role on the `admin` database can include privileges that apply to the other databases.
- `collection_name` - (Optional) Collection for which the role applies. You can specify a collection for the `read` and `readWrite` roles. If you do not specify a collection for `read` and `readWrite`, the role applies to all collections in the database (excluding some collections in the `system`. database).

» Labels

Containing key-value pairs that tag and categorize the database user. Each key and value has a maximum length of 255 characters.

- `key` - The key that you want to write.
- `value` - The value that you want to write.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The database user's name.

» Import

Database users can be imported using project ID and username, in the format `project_id-username-auth_database_name`, e.g.

```
$ terraform import mongodbatlas_database_user.my_user 1112222b3bf99403840e8934-my_user-admi
```

NOTE: Terraform will want to change the password after importing the user if a `password` argument is specified.

» mongodbatlas_project_ip_whitelist

`mongodbatlas_project_ip_whitelist` provides an IP Whitelist entry resource. The whitelist grants access from IPs, CIDRs or AWS Security Groups (if VPC Peering is enabled) to clusters within the Project.

NOTE: Groups and projects are synonymous terms. You may find `groupId` in the official documentation.

IMPORTANT: When you remove an entry from the whitelist, existing connections from the removed address(es) may remain open for a variable amount of time. How much time passes before Atlas closes the connection depends on several factors, including how the connection was established, the particular behavior of the application or driver using the address, and the connection protocol (e.g., TCP or UDP). This is particularly important to consider when changing an existing IP address or CIDR block as they cannot be updated via the Provider (comments can however), hence a change will force the destruction and recreation of entries.

» Example Usage

» Using CIDR Block

```
resource "mongodbatlas_project_ip_whitelist" "test" {
  project_id = "<PROJECT-ID>"
  cidr_block = "1.2.3.4/32"
  comment    = "cidr block for tf acc testing"
}
```

» Using IP Address

```
resource "mongodbatlas_project_ip_whitelist" "test" {
  project_id = "<PROJECT-ID>"
  ip_address = "2.3.4.5"
  comment    = "ip address for tf acc testing"
}
```

» Using an AWS Security Group

```
resource "mongodbatlas_network_container" "test" {
  project_id       = "<PROJECT-ID>"
  atlas_cidr_block = "192.168.208.0/21"
  provider_name    = "AWS"
  region_name      = "US_EAST_1"
```

```
}

resource "mongodbatlas_network_peering" "test" {
  project_id             = "<PROJECT-ID>"
  container_id           = mongodbatlas_network_container.test.container_id
  accepter_region_name   = "us-east-1"
  provider_name          = "AWS"
  route_table_cidr_block = "172.31.0.0/16"
  vpc_id                 = "vpc-0d93d6f69f1578bd8"
  aws_account_id         = "232589400519"
}

resource "mongodbatlas_project_ip_whitelist" "test" {
  project_id         = "<PROJECT-ID>"
  aws_security_group = "sg-0026348ec11780bd1"
  comment            = "TestAcc for awsSecurityGroup"

  depends_on = ["mongodbatlas_network_peering.test"]
}
```

IMPORTANT: In order to use AWS Security Group(s) VPC Peering must be enabled like above example.

» Argument Reference

- `project_id` - (Required) The ID of the project in which to add the whitelist entry.
- `aws_security_group` - (Optional) ID of the whitelisted AWS security group. Mutually exclusive with `cidr_block` and `ip_address`.
- `cidr_block` - (Optional) Whitelist entry in Classless Inter-Domain Routing (CIDR) notation. Mutually exclusive with `aws_security_group` and `ip_address`.
- `ip_address` - (Optional) Whitelisted IP address. Mutually exclusive with `aws_security_group` and `cidr_block`.
- `comment` - (Optional) Comment to add to the whitelist entry.

NOTE: One of the following attributes must set: `aws_security_group`, `cidr_block` or `ip_address`.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - Unique identifier used for terraform for internal manages and can be used to import.

» Import

IP Whitelist entries can be imported using the `project_id` and `cidr_block` or `ip_address`, e.g.

```
$ terraform import mongodbatlas_project_ip_whitelist.test 5d0f1f74cf09a29120e123cd-10.242.88
```

For more information see: MongoDB Atlas API Reference.

» mongodbatlas_cluster

`mongodbatlas_cluster` provides a Cluster resource. The resource lets you create, edit and delete clusters. The resource requires your Project ID.

NOTE: Groups and projects are synonymous terms. You may find group_id in the official documentation.

IMPORTANT:
• Free tier cluster creation (M0) is not supported via API or by this Provider.
• Shared tier clusters (M2, M5) cannot be upgraded to higher tiers via API or by this Provider.
• Changes to cluster configurations can affect costs. Before making changes, please see Billing.
• If your Atlas project contains a custom role that uses actions introduced in a specific MongoDB version, you cannot create a cluster with a MongoDB version less than that version unless you delete the custom role.

» Example Usage

» Example AWS cluster

```
resource "mongodbatlas_cluster" "cluster-test" {
  project_id   = "<YOUR-PROJECT-ID>"
  name         = "cluster-test"
  num_shards   = 1

  replication_factor           = 3
  provider_backup_enabled      = true
  auto_scaling_disk_gb_enabled = true
  mongo_db_major_version       = "4.0"

  //Provider Settings "block"
  provider_name               = "AWS"
  disk_size_gb                = 100
  provider_disk_iops          = 300
```

```
  provider_volume_type        = "STANDARD"
  provider_encrypt_ebs_volume = true
  provider_instance_size_name = "M40"
  provider_region_name        = "US_EAST_1"
}
```

» Example Azure cluster.

```
resource "mongodbatlas_cluster" "test" {
  project_id   = "<YOUR-PROJECT-ID>"
  name         = "test"
  num_shards   = 1

  replication_factor           = 3
  backup_enabled               = true
  auto_scaling_disk_gb_enabled = true
  mongo_db_major_version       = "4.0"

  //Provider Settings "block"
  provider_name               = "AZURE"
  provider_disk_type_name     = "P6"
  provider_instance_size_name = "M30"
  provider_region_name        = "US_EAST_2"
}
```

» Example GCP cluster

```
resource "mongodbatlas_cluster" "test" {
  project_id   = "<YOUR-PROJECT-ID>"
  name         = "test"
  num_shards   = 1

  replication_factor           = 3
  backup_enabled               = true
  auto_scaling_disk_gb_enabled = true
  mongo_db_major_version       = "4.0"

  //Provider Settings "block"
  provider_name               = "GCP"
  disk_size_gb                = 40
  provider_instance_size_name = "M30"
  provider_region_name        = "US_EAST_4"
}
```

» Example Multi Region cluster

```
resource "mongodbatlas_cluster" "cluster-test" {
  project_id               = "<YOUR-PROJECT-ID>"
  name                     = "cluster-test-multi-region"
  disk_size_gb             = 100
  num_shards               = 1
  provider_backup_enabled  = true
  cluster_type             = "REPLICASET"

  //Provider Settings "block"
  provider_name               = "AWS"
  provider_disk_iops          = 300
  provider_volume_type        = "STANDARD"
  provider_instance_size_name = "M10"

  replication_specs {
    num_shards = 1
    regions_config {
      region_name     = "US_EAST_1"
      electable_nodes = 3
      priority        = 7
      read_only_nodes = 0
    }
    regions_config {
      region_name     = "US_EAST_2"
      electable_nodes = 2
      priority        = 6
      read_only_nodes = 0
    }
    regions_config {
      region_name     = "US_WEST_1"
      electable_nodes = 2
      priority        = 5
      read_only_nodes = 2
    }
  }
}
```

» Example Global cluster

```
resource "mongodbatlas_cluster" "cluster-test" {
  project_id              = "<YOUR-PROJECT-ID>"
  name                    = "cluster-test-global"
  disk_size_gb            = 80
```

```
  num_shards              = 1
  provider_backup_enabled = true
  cluster_type            = "GEOSHARDED"

  //Provider Settings "block"
  provider_name               = "AWS"
  provider_disk_iops          = 240
  provider_volume_type        = "STANDARD"
  provider_instance_size_name = "M30"

  replication_specs {
    zone_name  = "Zone 1"
    num_shards = 2
    regions_config {
      region_name     = "US_EAST_1"
      electable_nodes = 3
      priority        = 7
      read_only_nodes = 0
    }
  }

  replication_specs {
    zone_name  = "Zone 2"
    num_shards = 2
    regions_config {
      region_name     = "EU_CENTRAL_1"
      electable_nodes = 3
      priority        = 7
      read_only_nodes = 0
    }
  }
}
```

» Example AWS Shared Tier cluster

```
resource "mongodbatlas_cluster" "cluster-test" {
  project_id              = "<YOUR-PROJECT-ID>"
  name                    = "cluster-test-global"
  //M2 must be 2, M5 must be 5
  disk_size_gb            = "2"

  //Provider Settings "block"
  provider_name = "TENANT"
  backing_provider_name = "AWS"
  provider_region_name = "US_EAST_1"
```

```
  provider_instance_size_name = "M2"

  //These must be the following values
  mongo_db_major_version = "4.2"
  auto_scaling_disk_gb_enabled = "false"
}
```

» Argument Reference

- `project_id` - (Required) The unique ID for the project to create the database user.
- `provider_name` - (Required) Cloud service provider on which the servers are provisioned.

  The possible values are:

  - `AWS` - Amazon AWS
  - `GCP` - Google Cloud Platform
  - `AZURE` - Microsoft Azure
  - `TENANT` - A multi-tenant deployment on one of the supported cloud service providers. Only valid when providerSettings.instanceSizeName is either M2 or M5.
- `name` - (Required) Name of the cluster as it appears in Atlas. Once the cluster is created, its name cannot be changed.
- `provider_instance_size_name` - (Required) Atlas provides different instance sizes, each with a default storage capacity and RAM size. The instance size you select is used for all the data-bearing servers in your cluster. See Create a Cluster `providerSettings.instanceSizeName` for valid values and default resources.

Note free tier (M0) creation is not supported by the Atlas API and hence not supported by this provider.

- `auto_scaling_disk_gb_enabled` - (Optional) Specifies whether disk auto-scaling is enabled. The default is true.
  - Set to `true` to enable disk auto-scaling.
  - Set to `false` to disable disk auto-scaling.
- `backup_enabled` - (Optional) Set to true to enable Atlas continuous backups for the cluster.

  Set to false to disable continuous backups for the cluster. Atlas deletes any stored snapshots. See the continuous backup Snapshot Schedule for more information.

You cannot enable continuous backups if you have an existing cluster in the project with Cloud Provider Snapshots enabled.

You cannot enable continuous backups for new AWS clusters. If backup is required for a new cluster use `provider_backup_enabled` to enable Cloud Provider Snapshots.

The default value is false. M10 and above only.

- `bi_connector` - (Optional) Specifies BI Connector for Atlas configuration on this cluster. BI Connector for Atlas is only available for M10+ clusters. See BI Connector below for more details.
- `cluster_type` - (Optional) Specifies the type of the cluster that you want to modify. You cannot convert a sharded cluster deployment to a replica set deployment.

  WHEN SHOULD YOU USE CLUSTERTYPE? When you set replication_specs, when you are deploying Global Clusters or when you are deploying non-Global replica sets and sharded clusters.

  Accepted values include:

  - `REPLICASET` Replica set
  - `SHARDED` Sharded cluster
  - `GEOSHARDED` Global Cluster
- `disk_size_gb` - (Optional - GCP/AWS Only) The size in gigabytes of the server's root volume. You can add capacity by increasing this number, up to a maximum possible value of 4096 (i.e., 4 TB). This value must be a positive integer.

  The minimum disk size for dedicated clusters is 10GB for AWS and GCP. If you specify diskSizeGB with a lower disk size, Atlas defaults to the minimum disk size value.
- `encryption_at_rest_provider` - (Optional) Set the Encryption at Rest parameter. Possible values are AWS, GCP, AZURE or NONE. Requires M10 or greater and for backup_enabled to be false or omitted.
- `labels` - (Optional) Array containing key-value pairs that tag and categorize the cluster. Each key and value has a maximum length of 255 characters. You cannot set the key `Infrastructure Tool`, it is used for internal purposes to track aggregate usage.
- `mongo_db_major_version` - (Optional) Version of the cluster to deploy. Atlas supports the following MongoDB versions for M10+ clusters: `3.6`, `4.0`, or `4.2`. You must set this value to `4.2` if `provider_instance_size_name` is either M2 or M5.
- `num_shards` - (Optional) Selects whether the cluster is a replica set or a sharded cluster. If you use the replicationSpecs parameter, you must set

num_shards.

- `pit_enabled` - (Optional) - Flag that indicates if the cluster uses Point-in-Time backups. If set to true, provider_backup_enabled must also be set to true.
- `provider_backup_enabled` - (Optional) Flag indicating if the cluster uses Cloud Provider Snapshots for backups.

  If true, the cluster uses Cloud Provider Snapshots for backups. If providerBackupEnabled and backupEnabled are false, the cluster does not use Atlas backups.

  You cannot enable cloud provider snapshots if you have an existing cluster in the project with Continuous Backups enabled.
- `backing_provider_name` - (Optional) Cloud service provider on which the server for a multi-tenant cluster is provisioned.

  This setting is only valid when providerSetting.providerName is TENANT and providerSetting.instanceSizeName is M2 or M5.

  The possible values are:

  - AWS - Amazon AWS
  - GCP - Google Cloud Platform
  - AZURE - Microsoft Azure
- `provider_disk_iops` - (Optional) The maximum input/output operations per second (IOPS) the system can perform. The possible values depend on the selected providerSettings.instanceSizeName and diskSizeGB.
- `provider_disk_type_name` - (Optional - Azure Only) Azure disk type of the server's root volume. If omitted, Atlas uses the default disk type for the selected providerSettings.instanceSizeName. Example disk types and associated storage sizes: PP4 - 32GB, P6 - 64GB, P10 - 128GB, P20 - 512GB, P30 - 1024GB, P40 - 2048GB, P50 - 4095GB. More information and the most update to date disk types/storage sizes can be located at https://docs.atlas.mongodb.com/reference/api/clusters-create-one/.
- `provider_encrypt_ebs_volume` - (Optional) If enabled, the Amazon EBS encryption feature encrypts the server's root volume for both data at rest within the volume and for data moving between the volume and the instance.
- `provider_region_name` - (Optional) Physical location of your MongoDB cluster. The region you choose can affect network latency for clients accessing your databases. Requires the Atlas Region name, see the reference list for AWS, GCP, Azure. Do not specify this field when creating a multi-region cluster using the replicationSpec document or a Global Cluster with the replicationSpecs array.

- `provider_volume_type` - (AWS - Optional) The type of the volume. The possible values are: `STANDARD` and `PROVISIONED`. `PROVISIONED` required if setting IOPS higher than the default instance IOPS.
- `replication_factor` - (Optional) Number of replica set members. Each member keeps a copy of your databases, providing high availability and data redundancy. The possible values are 3, 5, or 7. The default value is 3.
- `replication_specs` - (Optional) Configuration for cluster regions. See Replication Spec below for more details.

» BI Connector

Specifies BI Connector for Atlas configuration.

- `enabled` - (Optional) Specifies whether or not BI Connector for Atlas is enabled on the cluster.
  - Set to `true` to enable BI Connector for Atlas.
  - Set to `false` to disable BI Connector for Atlas.
- `read_preference` - (Optional) Specifies the read preference to be used by BI Connector for Atlas on the cluster. Each BI Connector for Atlas read preference contains a distinct combination of readPreference and readPreferenceTags options. For details on BI Connector for Atlas read preferences, refer to the BI Connector Read Preferences Table.
  - Set to "primary" to have BI Connector for Atlas read from the primary.
  - Set to "secondary" to have BI Connector for Atlas read from a secondary member. Default if there are no analytics nodes in the cluster.
  - Set to "analytics" to have BI Connector for Atlas read from an analytics node. Default if the cluster contains analytics nodes.

» Replication Spec

Configuration for cluster regions.

- `num_shards` - (Required) Number of shards to deploy in the specified zone.
- `id` - (Optional) Unique identifer of the replication document for a zone in a Global Cluster.
- `regions_config` - (Optional) Physical location of the region. Each regionsConfig document describes the region's priority in elections and the number and type of MongoDB nodes Atlas deploys to the region. You must order each regionsConfigs document by regionsConfig.priority, descending. See Region Config below for more details.
- `zone_name` - (Optional) Name for the zone in a Global Cluster.

» Region Config

Physical location of the region.

- `region_name` - (Optional) Name for the region specified.
- `electable_nodes` - (Optional) Number of electable nodes for Atlas to deploy to the region. Electable nodes can become the primary and can facilitate local reads.
- `priority` - (Optional) Election priority of the region. For regions with only read-only nodes, set this value to 0.
- `read_only_nodes` - (Optional) Number of read-only nodes for Atlas to deploy to the region. Read-only nodes can never become the primary, but can facilitate local-reads. Specify 0 if you do not want any read-only nodes in the region.
- `analytics_nodes` - (Optional) The number of analytics nodes for Atlas to deploy to the region. Analytics nodes are useful for handling analytic data such as reporting queries from BI Connector for Atlas. Analytics nodes are read-only, and can never become the primary.

  If you do not specify this option, no analytics nodes are deployed to the region.

» Advanced Configuration Options

NOTE: Prior to setting these options please ensure you read https://docs.atlas.mongodb.com/cluster-config/additional-options/.

Include **desired options** within advanced_configuration:

```
// Nest options within advanced_configuration
 advanced_configuration = {
   javascript_enabled                 = false
   minimum_enabled_tls_protocol       = "TLS1_2"
 }
```

- `fail_index_key_too_long` - (Optional) When true, documents can only be updated or inserted if, for all indexed fields on the target collection, the corresponding index entries do not exceed 1024 bytes. When false, mongod writes documents that exceed the limit but does not index them.
- `javascript_enabled` - (Optional) When true, the cluster allows execution of operations that perform server-side executions of JavaScript. When false, the cluster disables execution of those operations.

- `minimum_enabled_tls_protocol` - (Optional) Sets the minimum Transport Layer Security (TLS) version the cluster accepts for incoming connections.Valid values are:
  - TLS1_0
  - TLS1_1
  - TLS1_2
- `no_table_scan` - (Optional) When true, the cluster disables the execution of any query that requires a collection scan to return results. When false, the cluster allows the execution of those operations.
- `oplog_size_mb` - (Optional) The custom oplog size of the cluster. Without a value that indicates that the cluster uses the default oplog size calculated by Atlas.
- `sample_size_bi_connector` - (Optional) Number of documents per database to sample when gathering schema information. Defaults to 100. Available only for Atlas deployments in which BI Connector for Atlas is enabled.
- `sample_refresh_interval_bi_connector` - (Optional) Interval in seconds at which the mongosqld process re-samples data to create its relational schema. The default value is 300. The specified value must be a positive integer. Available only for Atlas deployments in which BI Connector for Atlas is enabled.

» Labels

Contains key-value pairs that tag and categorize the cluster. Each key and value has a maximum length of 255 characters.

- `key` - The key that you want to write.
- `value` - The value that you want to write.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `cluster_id` - The cluster ID.
- `mongo_db_version` - Version of MongoDB the cluster runs, in `major-version.minor-version` format.
- `id` - The Terraform's unique identifier used internally for state management.
- `mongo_uri` - Base connection string for the cluster. Atlas only displays this field after the cluster is operational, not while it builds the cluster.

- `mongo_uri_updated` - Lists when the connection string was last updated. The connection string changes, for example, if you change a replica set to a sharded cluster.
- `mongo_uri_with_options` - connection string for connecting to the Atlas cluster. Includes the replicaSet, ssl, and authSource query parameters in the connection string with values appropriate for the cluster.

  To review the connection string format, see the connection string format documentation. To add MongoDB users to a Atlas project, see Configure MongoDB Users.

  Atlas only displays this field after the cluster is operational, not while it builds the cluster.
- `paused` - Flag that indicates whether the cluster is paused or not.
- `srv_address` - Connection string for connecting to the Atlas cluster. The +srv modifier forces the connection to use TLS/SSL. See the mongoURI for additional options.
- `state_name` - Current state of the cluster. The possible states are:
  - IDLE
  - CREATING
  - UPDATING
  - DELETING
  - DELETED
  - REPAIRING

» Import

Clusters can be imported using project ID and cluster name, in the format `PROJECTID-CLUSTERNAME`, e.g.

```
$ terraform import mongodbatlas_cluster.my_cluster 1112222b3bf99403840e8934-Cluster0
```

See detailed information for arguments and attributes: MongoDB API Clusters

» mongodbatlas_network_container

`mongodbatlas_network_container` provides a Network Peering Container resource. The resource lets you create, edit and delete network peering containers. The resource requires your Project ID.

IMPORTANT: This resource creates one Network Peering container into which Atlas can deploy Network Peering connections. An Atlas project can have a maximum of one container for each cloud provider. You must have

either the Project Owner or Organization Owner role to successfully call this endpoint.

The following table outlines the maximum number of Network Peering containers per cloud provider: * Cloud Provider: GCP - Container Limit: One container per project. * Cloud Provider: AWS and Azure - Container Limit: One container per cloud provider region.

NOTE: Groups and projects are synonymous terms. You may find **group_id** in the official documentation.

» Example Usage

» Example with AWS.

```
  resource "mongodbatlas_network_container" "test" {
    project_id       = "<YOUR-PROJECT-ID>"
    atlas_cidr_block = "10.8.0.0/21"
    provider_name    = "AWS"
    region_name      = "US_EAST_1"
  }
```

» Example with GCP

```
resource "mongodbatlas_network_container" "test" {
  project_id       = "<YOUR-PROJECT-ID>"
  atlas_cidr_block = "10.8.0.0/21"
  provider_name    = "GCP"
}
```

» Example with Azure

```
resource "mongodbatlas_network_container" "test" {
  project_id       = "<YOUR-PROJECT-ID>"
  atlas_cidr_block = "10.8.0.0/21"
  provider_name    = "AZURE"
  region           = "US_EAST_2"
}
```

» Argument Reference

- `project_id` - (Required) The unique ID for the project to create the database user.

- `atlas_cidr_block` - (Required) CIDR block that Atlas uses for your clusters. Atlas uses the specified CIDR block for all other Network Peering connections created in the project. The Atlas CIDR block must be at least a /24 and at most a /21 in one of the following private networks.
- `provider_name` - (Optional) Cloud provider for this Network Peering connection. If omitted, Atlas sets this parameter to AWS.
- `region_name` - (Optional | AWS provider only) AWS region.
- `region` - (Optional | AZURE provider only) Azure region where the container resides.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `container_id` - The Network Peering Container ID.
- `id` - The Terraform's unique identifier used internally for state management.
- `region_name` - AWS region.
- `region` - Azure region where the container resides.
- `azure_subscription_id` - Unique identifer of the Azure subscription in which the VNet resides.
- `provisioned` - Indicates whether the project has Network Peering connections deployed in the container.
- `gcp_project_id` - Unique identifier of the GCP project in which the Network Peering connection resides.
- `network_name` - Name of the Network Peering connection in the Atlas project.
- `vpc_id` - Unique identifier of the project's VPC.
- `vnet_name` - The name of the Azure VNet. This value is null until you provision an Azure VNet in the container.

» Import

Clusters can be imported using project ID and network peering container id, in the format `PROJECTID-CONTAINER-ID`, e.g.

```
$ terraform import mongodbatlas_network_container.my_container 1112222b3bf99403840e8934-5cbf
```

See detailed information for arguments and attributes: MongoDB API Network Peering Container

» mongodbatlas_project

`mongodbatlas_project` provides a Project resource. This allows project to be created.

IMPORTANT WARNING: Changing the name of an existing Project in your Terraform configuration will result the destruction of that Project and related resources (including Clusters) and the re-creation of those resources. Terraform will inform you of the destroyed/created resources before applying so be sure to verify any change to your environment before applying.

» Example Usage

```
resource "mongodbatlas_project" "test" {
  name   = "project-name"
  org_id = "<ORG_ID>"

  teams {
    team_id    = "5e0fa8c99ccf641c722fe645"
    role_names = ["GROUP_OWNER"]

  }
  teams {
    team_id    = "5e1dd7b4f2a30ba80a70cd4rw"
    role_names = ["GROUP_READ_ONLY", "GROUP_DATA_ACCESS_READ_WRITE"]
  }
}
```

» Argument Reference

- `name` - (Required) The name of the project you want to create. (Cannot be changed via this Provider after creation.)
- `org_id` - (Required) The ID of the organization you want to create the project within.

» Teams

Teams attribute is optional

NOTE: Atlas limits the number of users to a maximum of 100 teams per project and a maximum of 250 teams per organization.

- `team_id` - (Required) The unique identifier of the team you want to associate with the project. The team and project must share the same parent organization.
- `role_names` - (Required) Each string in the array represents a project role you want to assign to the team. Every user associated with the team inherits these roles. You must specify an array even if you are only associating a single role with the team. The following are valid roles:
  - `GROUP_OWNER`
  - `GROUP_READ_ONLY`
  - `GROUP_DATA_ACCESS_ADMIN`
  - `GROUP_DATA_ACCESS_READ_WRITE`
  - `GROUP_DATA_ACCESS_READ_ONLY`
  - `GROUP_CLUSTER_MANAGER`

NOTE: Project created by API Keys must belong to an existing organization.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The project id.
- `created` - The ISO-8601-formatted timestamp of when Atlas created the project..
- `cluster_count` - The number of Atlas clusters deployed in the project..

» Import

Project must be imported using project ID, e.g.

```
$ terraform import mongodbatlas_project.my_project 5d09d6a59ccf6445652a444a
```

For more information see: MongoDB Atlas API Reference. - and MongoDB Atlas API - Teams Documentation for more information.

» mongodbatlas_cloud_provider_snapshot

`mongodbatlas_cloud_provider_snapshot` provides a resource to take a cloud provider snapshot on demand. On-demand snapshots happen immediately, unlike scheduled snapshots which occur at regular intervals. If there is already an on-demand snapshot with a status of queued or inProgress, you must wait until Atlas has completed the on-demand snapshot before taking another.

NOTE: Groups and projects are synonymous terms. You may find `groupId` in the official documentation.

» Example Usage

```
resource "mongodbatlas_cluster" "my_cluster" {
  project_id   = "5cf5a45a9ccf6400e60981b6"
  name         = "MyCluster"
  disk_size_gb = 5

//Provider Settings "block"
  provider_name               = "AWS"
  provider_region_name        = "EU_WEST_2"
  provider_instance_size_name = "M10"
  provider_backup_enabled     = true   // enable cloud provider snapshots
  provider_disk_iops          = 100
  provider_encrypt_ebs_volume = false
}

resource "mongodbatlas_cloud_provider_snapshot" "test" {
  project_id        = mongodbatlas_cluster.my_cluster.project_id
  cluster_name      = mongodbatlas_cluster.my_cluster.name
  description       = "myDescription"
  retention_in_days = 1
}

resource "mongodbatlas_cloud_provider_snapshot_restore_job" "test" {
  project_id      = mongodbatlas_cloud_provider_snapshot.test.project_id
  cluster_name    = mongodbatlas_cloud_provider_snapshot.test.cluster_name
  snapshot_id     = mongodbatlas_cloud_provider_snapshot.test.snapshot_id
  delivery_type = {
    download = true
  }
}
```

» Argument Reference

- `project_id` - (Required) The unique identifier of the project for the Atlas cluster.
- `cluster_name` - (Required) The name of the Atlas cluster that contains the snapshots you want to retrieve.
- `description` - (Required) Description of the on-demand snapshot.
- `retention_in_days` - (Required) The number of days that Atlas should retain the on-demand snapshot. Must be at least 1.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `snapshot_id` - Unique identifier of the snapshot.
- `id` - Unique identifier used for terraform for internal manages.
- `created_at` - UTC ISO 8601 formatted point in time when Atlas took the snapshot.
- `description` - Description of the snapshot. Only present for on-demand snapshots.
- `expires_at` - UTC ISO 8601 formatted point in time when Atlas will delete the snapshot.
- `master_key_uuid` - Unique ID of the AWS KMS Customer Master Key used to encrypt the snapshot. Only visible for clusters using Encryption at Rest via Customer KMS.
- `mongod_version` - Version of the MongoDB server.
- `snapshot_type` - Specified the type of snapshot. Valid values are onDemand and scheduled.
- `status` - Current status of the snapshot. One of the following values will be returned: queued, inProgress, completed, failed.
- `storage_size_bytes` - Specifies the size of the snapshot in bytes.
- `type` - Specifies the type of cluster: replicaSet or shardedCluster.

» Import

Cloud Provider Snapshot entries can be imported using project project_id, cluster_name and snapshot_id (Unique identifier of the snapshot), in the format `PROJECTID-CLUSTERNAME-SNAPSHOTID`, e.g.

```
$ terraform import mongodbatlas_cloud_provider_snapshot.test 5d0f1f73cf09a29120e173cf-MyClus
```

For more information see: MongoDB Atlas API Reference.

» mongodbatlas_cloud_provider_snapshot_restore_job

`mongodbatlas_cloud_provider_snapshot_restore_job` provides a resource to create a new restore job from a cloud provider snapshot of a specified cluster. The restore job can be one of two types: * **automated:** Atlas automatically restores the snapshot with snapshotId to the Atlas cluster with name targetClusterName in the Atlas project with targetGroupId.

- **download:** Atlas provides a URL to download a .tar.gz of the snapshot with snapshotId. The contents of the archive contain the data files for your Atlas cluster.

NOTE: Groups and projects are synonymous terms. You may find `groupId` in the official documentation.

» Example Usage

» Example automated delivery type.

```
resource "mongodbatlas_cluster" "my_cluster" {
  project_id   = "5cf5a45a9ccf6400e60981b6"
  name         = "MyCluster"
  disk_size_gb = 5

//Provider Settings "block"
  provider_name               = "AWS"
  provider_region_name        = "EU_WEST_2"
  provider_instance_size_name = "M10"
  provider_backup_enabled     = true   // enable cloud provider snapshots
  provider_disk_iops          = 100
  provider_encrypt_ebs_volume = false
}

resource "mongodbatlas_cloud_provider_snapshot" "test" {
  project_id        = mongodbatlas_cluster.my_cluster.project_id
  cluster_name      = mongodbatlas_cluster.my_cluster.name
  description       = "myDescription"
  retention_in_days = 1
}

resource "mongodbatlas_cloud_provider_snapshot_restore_job" "test" {
  project_id      = mongodbatlas_cloud_provider_snapshot.test.project_id
  cluster_name    = mongodbatlas_cloud_provider_snapshot.test.cluster_name
  snapshot_id     = mongodbatlas_cloud_provider_snapshot.test.snapshot_id
  delivery_type   = {
    automated           = true
    target_cluster_name = "MyCluster"
    target_project_id   = "5cf5a45a9ccf6400e60981b6"
  }
  depends_on = ["mongodbatlas_cloud_provider_snapshot.test"]
}
```

» Example download delivery type.

```
resource "mongodbatlas_cluster" "my_cluster" {
  project_id   = "5cf5a45a9ccf6400e60981b6"
```

```
  name         = "MyCluster"
  disk_size_gb = 5

//Provider Settings "block"
  provider_name               = "AWS"
  provider_region_name        = "EU_WEST_2"
  provider_instance_size_name = "M10"
  provider_backup_enabled     = true   // enable cloud provider snapshots
  provider_disk_iops          = 100
  provider_encrypt_ebs_volume = false
}

resource "mongodbatlas_cloud_provider_snapshot" "test" {
  project_id        = mongodbatlas_cluster.my_cluster.project_id
  cluster_name      = mongodbatlas_cluster.my_cluster.name
  description       = "myDescription"
  retention_in_days = 1
}

resource "mongodbatlas_cloud_provider_snapshot_restore_job" "test" {
  project_id      = mongodbatlas_cloud_provider_snapshot.test.project_id
  cluster_name    = mongodbatlas_cloud_provider_snapshot.test.cluster_name
  snapshot_id     = mongodbatlas_cloud_provider_snapshot.test.snapshot_id
  delivery_type = {
    download = true
  }
}
```

» Argument Reference

- `project_id` - (Required) The unique identifier of the project for the Atlas cluster whose snapshot you want to restore.
- `cluster_name` - (Required) The name of the Atlas cluster whose snapshot you want to restore.
- `snapshot_id` - (Required) Unique identifier of the snapshot to restore.
- `delivery_type` - (Required) Type of restore job to create. Possible values are: **download** or **automated**, only one must be set it in `true`.

» Download

Atlas provides a URL to download a .tar.gz of the snapshot with snapshotId.

» Automated

Atlas automatically restores the snapshot with snapshotId to the Atlas cluster with name targetClusterName in the Atlas project with targetGroupId. if you want to use automated delivery type, you must to set the following arguments:

- `target_cluster_name` - (Required) Name of the target Atlas cluster to which the restore job restores the snapshot. Only required if deliveryType is automated.
- `target_group_id` - (Required) Unique ID of the target Atlas project for the specified targetClusterName. Only required if deliveryType is automated.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `snapshot_restore_job_id` - The unique identifier of the restore job.
- `cancelled` - Indicates whether the restore job was canceled.
- `created_at` - UTC ISO 8601 formatted point in time when Atlas created the restore job.
- `delivery_type` - Type of restore job to create. Possible values are: automated and download.
- `delivery_url` - One or more URLs for the compressed snapshot files for manual download. Only visible if deliveryType is download.
- `expired` - Indicates whether the restore job expired.
- `expires_at` - UTC ISO 8601 formatted point in time when the restore job expires.
- `finished_at` - UTC ISO 8601 formatted point in time when the restore job completed.
- `id` - The Terraform's unique identifier used internally for state management.
- `links` - One or more links to sub-resources and/or related resources. The relations between URLs are explained in the Web Linking Specification.
- `snapshot_id` - Unique identifier of the source snapshot ID of the restore job.
- `target_group_id` - Name of the target Atlas project of the restore job. Only visible if deliveryType is automated.
- `target_cluster_name` - Name of the target Atlas cluster to which the restore job restores the snapshot. Only visible if deliveryType is automated.
- `timestamp` - Timestamp in ISO 8601 date and time format in UTC when the snapshot associated to snapshotId was taken.

» Import

Cloud Provider Snapshot Restore Job entries can be imported using project project_id, cluster_name and snapshot_id (Unique identifier of the snapshot), in the format `PROJECTID-CLUSTERNAME-JOBID`, e.g.

```
$ terraform import mongodbatlas_cloud_provider_snapshot_restore_job.test 5cf5a45a9ccf6400e60
```

For more information see: MongoDB Atlas API Reference.

» mongodbatlas_encryption_at_rest

`mongodbatlas_encryption_at_rest` Atlas encrypts your data at rest using encrypted storage media. Using keys you manage with AWS KMS, Atlas encrypts your data a second time when it writes it to the MongoDB encrypted storage engine. You can use the following clouds: AWS CMK, AZURE KEY VAULT and GOOGLE KEY VAULT to encrypt the MongoDB master encryption keys.

NOTE: Groups and projects are synonymous terms. You may find `groupId` in the official documentation.

» Example Usage

```
resource "mongodbatlas_encryption_at_rest" "test" {
  project_id = "<PROJECT-ID>"

  aws_kms = {
    enabled                = true
    access_key_id          = "AKIAIOSFODNN7EXAMPLE"
    secret_access_key      = "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
    customer_master_key_id = "030gce02-586d-48d2-a966-05ea954fde0g"
    region                 = "US_EAST_1"
  }

  azure_key_vault = {
    enabled             = true
    client_id           = "g54f9e2-89e3-40fd-8188-EXAMPLEID"
    azure_environment   = "AZURE"
    subscription_id     = "0ec944e3-g725-44f9-a147-EXAMPLEID"
    resource_group_name = "ExampleRGName"
    key_vault_name      = "EXAMPLEKeyVault"
    key_identifier      = "https://EXAMPLEKeyVault.vault.azure.net/keys/EXAMPLEKey/d891821e3
    secret              = "EXAMPLESECRET"
    tenant_id           = "e8e4b6ba-ff32-4c88-a9af-EXAMPLEID"
  }
```

```
  google_cloud_kms = {
    enabled                 = true
    service_account_key     = "{\"type\": \"service_account\",\"project_id\": \"my-project-c
    key_version_resource_id = "projects/my-project-common-0/locations/us-east4/keyRings/my-k
  }
}
```

» Argument Reference

- `project_id` - (Required) The unique identifier for the project.
- `aws_kms` - (Required) Specifies AWS KMS configuration details and whether Encryption at Rest is enabled for an Atlas project.
- `azure_key_vault` - (Required) Specifies Azure Key Vault configuration details and whether Encryption at Rest is enabled for an Atlas project.
- `google_cloud_kms` - (Required) Specifies GCP KMS configuration details and whether Encryption at Rest is enabled for an Atlas project.

» aws_kms

- `enabled` - Specifies whether Encryption at Rest is enabled for an Atlas project, To disable Encryption at Rest, pass only this parameter with a value of false, When you disable Encryption at Rest, Atlas also removes the configuration details.
- `access_key_id` - The IAM access key ID with permissions to access the customer master key specified by customerMasterKeyID.
- `secret_access_key` - The IAM secret access key with permissions to access the customer master key specified by customerMasterKeyID.
- `customer_master_key_id` - The AWS customer master key used to encrypt and decrypt the MongoDB master keys.
- `region` - The AWS region in which the AWS customer master key exists: CA_CENTRAL_1, US_EAST_1, US_EAST_2, US_WEST_1, US_WEST_2, SA_EAST_1

» azure_key_vault

- `enabled` - Specifies whether Encryption at Rest is enabled for an Atlas project. To disable Encryption at Rest, pass only this parameter with a value of false. When you disable Encryption at Rest, Atlas also removes the configuration details.
- `client_id` - The client ID, also known as the application ID, for an Azure application associated with the Azure AD tenant.

- `azure_environment` - The Azure environment where the Azure account credentials reside. Valid values are the following: AZURE, AZURE_CHINA, AZURE_GERMANY
- `subscription_id` - The unique identifier associated with an Azure subscription.
- `resource_group_name` - The name of the Azure Resource group that contains an Azure Key Vault.
- `key_vault_name` - The name of an Azure Key Vault containing your key.
- `key_identifier` - The unique identifier of a key in an Azure Key Vault.
- `secret` - The secret associated with the Azure Key Vault specified by azureKeyVault.tenantID.
- `tenant_id` - The unique identifier for an Azure AD tenant within an Azure subscription.

» google_cloud_kms

- `enabled` - Specifies whether Encryption at Rest is enabled for an Atlas project. To disable Encryption at Rest, pass only this parameter with a value of false. When you disable Encryption at Rest, Atlas also removes the configuration details.
- `service_account_key` - String-formatted JSON object containing GCP KMS credentials from your GCP account.
- `key_version_resource_id` - The Key Version Resource ID from your GCP account.

For more information see: MongoDB Atlas API Reference.

» mongodbatlas_network_peering

`mongodbatlas_network_peering` provides a Network Peering Connection resource. The resource lets you create, edit and delete network peering connections. The resource requires your Project ID. Ensure you have first created a Network Container. See the network_container resource and examples below.

GCP AND AZURE ONLY: You must enable Connect via Peering Only mode to use network peering.

AZURE ONLY: To create the peering request with an Azure VNET, you must grant Atlas the following permissions on the virtual network. Microsoft.Network/virtualNetworks/virtualNetworkPeerings/read Microsoft.Network/virtualNetworks/virtualNetworkPeerings/write Microsoft.Network/virtualNetworks/virtualNetworkPeerings/delete Microsoft.Network/virtualNetworks/peer/act
For more information see https://docs.atlas.mongodb.com/security-vpc-peering/

Create a Whitelist: Ensure you whitelist the private IP ranges of the subnets in which your application is hosted in order to connect to your Atlas cluster. See the project_ip_whitelist resource.

NOTE: Groups and projects are synonymous terms. You may find **group_id** in the official documentation.

» Example Usage

» Global configuration for the following examples

```
locals {
  project_id        = <your-project-id>

  # needed for GCP only
  google_project_id = <your-google-project-id>
}
```

» Example with AWS.

```
resource "mongodbatlas_network_container" "test" {
  project_id       = local.project_id
  atlas_cidr_block = "10.8.0.0/21"
  provider_name    = "AWS"
  region_name      = "US_EAST_1"
}

resource "mongodbatlas_network_peering" "test" {
  accepter_region_name   = "us-east-1"
  project_id             = local.project_id
  container_id           = "507f1f77bcf86cd799439011"
  provider_name          = "AWS"
  route_table_cidr_block = "192.168.0.0/24"
  vpc_id                 = "vpc-abc123abc123"
  aws_account_id         = "abc123abc123"
}

# the following assumes an AWS provider is configured
resource "aws_vpc_peering_connection_accepter" "peer" {
  vpc_peering_connection_id = "${mongodbatlas_network_peering.test.connection_id}"
  auto_accept = true
}
```

» Example with GCP

```
resource "mongodbatlas_network_container" "test" {
  project_id       = local.project_id
  atlas_cidr_block = "192.168.192.0/18"
  provider_name    = "GCP"
}

resource "mongodbatlas_private_ip_mode" "my_private_ip_mode" {
  project_id = local.project_id
  enabled    = true
}

resource "mongodbatlas_network_peering" "test" {
  project_id     = local.project_id
  container_id   = mongodbatlas_network_container.test.container_id
  provider_name  = "GCP"
  network_name   = "myNetWorkPeering"
  gcp_project_id = local.google_project_id

  depends_on = [mongodbatlas_private_ip_mode.my_private_ip_mode]
}

resource "google_compute_network" "vpc_network" {
  name = "vpcnetwork"
}

resource "google_compute_network_peering" "gcp_main_atlas_peering" {
  name         = "atlas-gcp-main"
  network      = google_compute_network.vpc_network.self_link
  peer_network = "projects/${mongodbatlas_network_peering.test.atlas_gcp_project_id}/global/
}
```

» Example with Azure

```
resource "mongodbatlas_network_container" "test" {
  project_id       = local.project_id
  atlas_cidr_block = "10.8.0.0/21"
  provider_name    = "AZURE"
  region           = "US_WEST"
}

resource "mongodbatlas_private_ip_mode" "my_private_ip_mode" {
  project_id = "${mongodbatlas_project.my_project.id}"
  enabled    = true
```

```
}

resource "mongodbatlas_network_peering" "test" {
  project_id            = local.project_id
  atlas_cidr_block      = "10.8.0.0/21"
  container_id          = mongodbatlas_network_container.test.container_id
  provider_name         = "AZURE"
  azure_directory_id    = "35039750-6ebd-4ad5-bcfe-cb4e5fc2d915"
  azure_subscription_id = "g893dec2-d92e-478d-bc50-cf99d31bgeg9"
  resource_group_name   = "atlas-azure-peering"
  vnet_name             = "azure-peer"

  depends_on = [mongodbatlas_private_ip_mode.my_private_ip_mode]
}
```

» Argument Reference

- `project_id` - (Required) The unique ID for the project to create the database user.
- `container_id` - (Required) Unique identifier of the Atlas VPC container for the region. You can create an Atlas VPC container using the Create Container endpoint. You cannot create more than one container per region. To retrieve a list of container IDs, use the Get list of VPC containers endpoint.
- `provider_name` - (Required) Cloud provider for this VPC peering connection. (Possible Values `AWS`, `AZURE`, `GCP`).
- `accepter_region_name` - (Optional | **AWS Required**) Specifies the region where the peer VPC resides. For complete lists of supported regions, see Amazon Web Services.
- `aws_account_id` - (Optional | **AWS Required**) Account ID of the owner of the peer VPC.
- `route_table_cidr_block` - (Optional | **AWS Required**) Peer VPC CIDR block or subnet.
- `vpc_id` - (Optional | **AWS Required**) Unique identifier of the peer VPC.
- `atlas_cidr_block` - (Optional | **AZURE Required**) Unique identifier for an Azure AD directory.
- `azure_directory_id` - (Optional | **AZURE Required**) Unique identifier for an Azure AD directory.
- `azure_subscription_id` - (Optional | **AZURE Required**) Unique identifer of the Azure subscription in which the VNet resides.
- `resource_group_name` - (Optional | **AZURE Required**) Name of your Azure resource group.
- `vnet_name` - (Optional | **AZURE Required**) Name of your Azure VNet.
- `gcp_project_id` - (Optinal | **GCP Required**) GCP project ID of the owner of the network peer.

- `network_name` - (Optional | **GCP Required**) Name of the network peer to which Atlas connects.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `peer_id` - The Network Peering Container ID.
- `id` - The Terraform's unique identifier used internally for state management.
- `connection_id` - Unique identifier for the peering connection.
- `accepter_region_name` - Specifies the region where the peer VPC resides. For complete lists of supported regions, see Amazon Web Services.
- `aws_account_id` - Account ID of the owner of the peer VPC.
- `provider_name` - Cloud provider for this VPC peering connection. If omitted, Atlas sets this parameter to AWS. (Possible Values `AWS`, `AZURE`, `GCP`).
- `route_table_cidr_block` - Peer VPC CIDR block or subnet.
- `vpc_id` - Unique identifier of the peer VPC.
- `error_state_name` - Error state, if any. The VPC peering connection error state value can be one of the following: `REJECTED`, `EXPIRED`, `INVALID_ARGUMENT`.
- `status_name` - The VPC peering connection status value can be one of the following: `INITIATING`, `PENDING_ACCEPTANCE`, `FAILED`, `FINALIZING`, `AVAILABLE`, `TERMINATING`.
- `atlas_cidr_block` - Unique identifier for an Azure AD directory.
- `azure_directory_id` - Unique identifier for an Azure AD directory.
- `azure_subscription_id` - Unique identifer of the Azure subscription in which the VNet resides.
- `resource_group_name` - Name of your Azure resource group.
- `vnet_name` - Name of your Azure VNet.
- `error_state` - Description of the Atlas error when `status` is `Failed`, Otherwise, Atlas returns `null`.
- `status` - Status of the Atlas network peering connection: `ADDING_PEER`, `AVAILABLE`, `FAILED`, `DELETING`, `WAITING_FOR_USER`.
- `gcp_project_id` - GCP project ID of the owner of the network peer.
- `atlas_gcp_project_id` - The Atlas GCP Project ID for the GCP VPC used by your atlas cluster that it is need to set up the reciprocal connection.
- `atlas_vpc_name` - The Atlas VPC Name is used by your atlas clister that it is need to set up the reciprocal connection.
- `network_name` - Name of the network peer to which Atlas connects.
- `error_message` - When `"status" : "FAILED"`, Atlas provides a description of the error.

» Import

Clusters can be imported using project ID and network peering peering id, in the format `PROJECTID-PEERID-PROVIDERNAME`, e.g.

```
$ terraform import mongodbatlas_network_peering.my_peering 1112222b3bf99403840e8934-5cbf563c
```

See detailed information for arguments and attributes: MongoDB API Network Peering Connection

» mongodbatlas_private_ip_mode

`mongodbatlas_private_ip_mode` provides a Private IP Mode resource. This allows one to enable/disable Connect via Peering Only mode for a MongoDB Atlas Project.

IMPORTANT:
What is Connect via Peering Only Mode?
Connect via Peering Only mode prevents clusters in an Atlas project from connecting to any network destination other than an Atlas Network Peer. Connect via Peering Only mode applies only to **GCP** and **Azure-backed** dedicated clusters. This setting disables the ability to:

- Deploy non-GCP or Azure-backed dedicated clusters in an Atlas project, and
- Use MongoDB Stitch with dedicated clusters in an Atlas project.

NOTE: You should create one private_ip_mode per project.

» Example Usage

```
resource "mongodbatlas_private_ip_mode" "my_private_ip_mode" {
    project_id = "<YOUR PROJECT ID>"
    enabled    = true
}
```

» Argument Reference

- `project_id` - (Required) The unique ID for the project to enable Only Private IP Mode.
- `enabled` - (Required) Indicates whether Connect via Peering Only mode is enabled or disabled for an Atlas project.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The project id.

» Import

Project must be imported using project ID, e.g.

```
$ terraform import mongodbatlas_private_ip_mode.my_private_ip_mode 5d09d6a59ccf6445652a444a
```

For more information see: MongoDB Atlas API Reference.

» mongodbatlas_maintenance_window

`mongodbatlas_maintenance_window` provides a resource to schedule a maintenance window for your MongoDB Atlas Project and/or set to defer a scheduled maintenance up to two times.

NOTE: Groups and projects are synonymous terms. You may find `groupId` in the official documentation.

» Maintenance Window Considerations:

- Urgent Maintenance Activities Cannot Wait: Urgent maintenance activities such as security patches cannot wait for your chosen window. Atlas will start those maintenance activities when needed.

Once maintenance is scheduled for your cluster, you cannot change your maintenance window until the current maintenance efforts have completed. - Maintenance Requires Replica Set Elections: Atlas performs maintenance the same way as the manual maintenance procedure. This requires at least one replica set election during the maintenance window per replica set. - Maintenance Starts As Close to the Hour As Possible: Maintenance always begins as close to the scheduled hour as possible, but in-progress cluster updates or expected system issues could delay the start time.

» Example Usage

```
resource "mongodbatlas_maintenance_window" "test" {
  project_id  = "<your-project-id>"
  day_of_week = 3
  hour_of_day = 4
```

```
}

resource "mongodbatlas_maintenance_window" "test" {
  project_id = "<your-project-id>"
  defer      = true
}
```

» Argument Reference

- `project_id` - The unique identifier of the project for the Maintenance Window.
- `day_of_week` - Day of the week when you would like the maintenance window to start as a 1-based integer: S=1, M=2, T=3, W=4, T=5, F=6, S=7.
- `hour_of_day` - Hour of the day when you would like the maintenance window to start. This parameter uses the 24-hour clock, where midnight is 0, noon is 12 (Time zone is UTC).
- `start_asap` - Flag indicating whether project maintenance has been directed to start immediately. If you request that maintenance begin immediately, this field returns true from the time the request was made until the time the maintenance event completes.
- `number_of_deferrals` - Number of times the current maintenance event for this project has been deferred, you can set a maximum of 2 deferrals.
- `defer` - Defer maintenance for the given project for one week.

NOTE: The `start_asap` attribute can't be used because of breaks the Terraform flow, but you can enable via API.

» Import

Maintenance Window entries can be imported using project project_id, in the format `PROJECTID`, e.g.

```
$ terraform import mongodbatlas_maintenance_window.test 5d0f1f73cf09a29120e173cf
```

For more information see: MongoDB Atlas API Reference.

» mongodbatlas_auditing

`mongodbatlas_auditing` provides an Auditing resource. This allows auditing to be created.

» Example Usage

```
resource "mongodbatlas_auditing" "test" {
        project_id                 = "<project-id>"
        audit_filter               = "{ 'atype': 'authenticate', 'param': {   'user': 'audi
        audit_authorization_success = false
        enabled                    = true
    }
```

» Argument Reference

- `project_id` - (Required) The unique ID for the project to configure auditing.
- `audit_filter` - Indicates whether the auditing system captures successful authentication attempts for audit filters using the ”atype” : ”authCheck” auditing event. For more information, see auditAuthorizationSuccess
- `audit_authorization_success` - JSON-formatted audit filter used by the project
- `enabled` - Denotes whether or not the project associated with the {project_id} has database auditing enabled.

NOTE: Auditing created by API Keys must belong to an existing organization.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `configuration_type` - Denotes the configuration method for the audit filter. Possible values are:
  - NONE - auditing not configured for the project.
  - FILTER_BUILDER - auditing configured via Atlas UI filter builder.
  - FILTER_JSON - auditing configured via Atlas custom filter or API.

» Import

Auditing must be imported using auditing ID, e.g.

```
$ terraform import mongodbatlas_auditing.my_auditing 5d09d6a59ccf6445652a444a
```

For more information see: MongoDB Atlas API Reference.

» mongodbatlas_teams

`mongodbatlas_teams` provides a Team resource. The resource lets you create, edit and delete Teams. Also, Teams can be assigned to multiple projects, and team members' access to the project is determined by the team's project role.

> **IMPORTANT:** MongoDB Atlas Team limits: max 250 teams in an organization and max 100 teams per project.

NOTE: Groups and projects are synonymous terms. You may find group_id in the official documentation.

MongoDB Atlas Team limits: max 250 teams in an organization and max 100 teams per project.

» Example Usage

```
resource "mongodbatlas_teams" "test" {
  org_id     = "<ORGANIZATION-ID>"
  name       = "myNewTeam"
  usernames  = ["user1", "user2", "user3"]
}
```

» Argument Reference

- `org_id` - (Required) The unique identifier for the organization you want to associate the team with.
- `name` - (Required) The name of the team you want to create.
- `usernames` - (Required) You can only add Atlas users who are part of the organization. Users who have not accepted an invitation to join the organization cannot be added as team members. There is a maximum of 250 Atlas users per team.

» Attributes Reference

In addition to all arguments above, the following attributes are exported: * `id` - The Terraform's unique identifier used internally for state management. * `team_id` - The unique identifier for the team.

» Import

Teams can be imported using the organization ID and team id, in the format ORGID-TEAMID, e.g.

```
$ terraform import mongodbatlas_teams.my_team 1112222b3bf99403840e8934-1112222b3bf99403840e8
```

See detailed information for arguments and attributes: MongoDB API Teams

» mongodbatlas_custom_db_role

`mongodbatlas_custom_db_role` provides a Custom DB Role resource. The customDBRoles resource lets you retrieve, create and modify the custom MongoDB roles in your cluster. Use custom MongoDB roles to specify custom sets of actions which cannot be described by the built-in Atlas database user privileges.

IMPORTANT Custom roles cannot use actions unavailable to any cluster version in your project. Custom roles are defined at the project level, and must be compatible with each MongoDB version used by your project's clusters. If you have a cluster in your project with MongoDB 3.4, you cannot create a custom role that uses actions introduced in MongoDB 3.6, such as useUUID.

NOTE: Groups and projects are synonymous terms. You may find group_id in the official documentation.

» Example Usage

```
resource "mongodbatlas_custom_db_role" "test_role" {
  project_id = "<PROJECT-ID>"
  role_name  = "myCustomRole"

  actions {
    action = "UPDATE"
    resources {
      collection_name = ""
      database_name   = "anyDatabase"
    }
  }
  actions {
    action = "INSERT"
    resources {
      collection_name = ""
      database_name   = "anyDatabase"
    }
  }
  actions {
    action = "REMOVE"
    resources {
      collection_name = ""
      database_name   = "anyDatabase"
```

```
    }
  }
}
```

» Example Usage with inherited roles

```
resource "mongodbatlas_custom_db_role" "inherited_role_one" {
  project_id = "<PROJECT-ID>"
  role_name  = "insertRole"

  actions {
    action = "INSERT"
    resources {
      collection_name = ""
      database_name   = "anyDatabase"
    }
  }
}

resource "mongodbatlas_custom_db_role" "inherited_role_two" {
  project_id = "${mongodbatlas_custom_db_role.inherited_role_one.project_id}"
  role_name  = "statusServerRole"

  actions {
    action = "SERVER_STATUS"
    resources {
      cluster = true
    }
  }
}

resource "mongodbatlas_custom_db_role" "test_role" {
  project_id = "${mongodbatlas_custom_db_role.inherited_role_one.project_id}"
  role_name  = "myCustomRole"

  actions {
    action = "UPDATE"
    resources {
      collection_name = ""
      database_name   = "anyDatabase"
    }
  }
  actions {
    action = "REMOVE"
    resources {
```

```
      collection_name = ""
      database_name   = "anyDatabase"
    }
  }

  inherited_roles {
    role_name     = "${mongodbatlas_custom_db_role.inherited_role_one.role_name}"
    database_name = "admin"
  }

  inherited_roles {
    role_name     = "${mongodbatlas_custom_db_role.inherited_role_two.role_name}"
    database_name = "admin"
  }
}
```

» Argument Reference

- `project_id` - (Required) The unique ID for the project to create the database user.
- `role_name` - (Required) Name of the custom role.

  IMPORTANT The specified role name can only contain letters, digits, underscores, and dashes. Additionally, you cannot specify a role name which meets any of the following criteria:

  - Is a name already used by an existing custom role in the project
  - Is a name of any of the built-in roles
  - Is `atlasAdmin`
  - Starts with `xgen-`

» Actions

Each object in the actions array represents an individual privilege action granted by the role. It is an required field.

- `action` - (Required) Name of the privilege action. For a complete list of actions available in the Atlas API, see Custom Role Actions -> **Note**: The privilege actions available to the Custom Roles API resource represent a subset of the privilege actions available in the Atlas Custom Roles UI.
- `resources` - (Required) Contains information on where the action is granted. Each object in the array either indicates a database and collection on which the action is granted, or indicates that the action is granted on the cluster resource.

- `resources.#.collection` - (Optional) Collection on which the action is granted. If this value is an empty string, the action is granted on all collections within the database specified in the actions.resources.db field.

  NOTE This field is mutually exclusive with the `actions.resources.cluster` field.

- `resources.#.database_name` Database on which the action is granted.

  NOTE This field is mutually exclusive with the `actions.resources.cluster` field.

- `resources.#.cluster` (Optional) Set to true to indicate that the action is granted on the cluster resource.

  NOTE This field is mutually exclusive with the `actions.resources.collection` and `actions.resources.db fields`.

» Inherited Roles

Each object in the inheritedRoles array represents a key-value pair indicating the inherited role and the database on which the role is granted. It is an optional field.

- `database_name` (Required) Database on which the inherited role is granted.

  NOTE This value should be admin for all roles except read and readWrite.

- `role_name` (Required) Name of the inherited role. This can either be another custom role or a built-in role.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - Unique identifier used for terraform for internal manages and can be used to import.

» Import

Database users can be imported using project ID and username, in the format `PROJECTID-ROLENAME`, e.g.

```
$ terraform import mongodbatlas_custom_db_role.my_role 1112222b3bf99403840e8934-MyCustomRole
```

For more information see: MongoDB Atlas API Reference.

» mongodbatlas_global_cluster_config

`mongodbatlas_global_cluster_config` provides a Global Cluster Configuration resource.

NOTE: Groups and projects are synonymous terms. You may find group_id in the official documentation.

» Examples Usage

» Example Global cluster

```
resource "mongodbatlas_cluster" "test" {
    project_id              = "<YOUR-PROJECT-ID>"
    name                    = "<CLUSTER-NAME>"
    disk_size_gb            = 80
    backup_enabled          = false
    provider_backup_enabled = true
    cluster_type            = "GEOSHARDED"

    //Provider Settings "block"
    provider_name               = "AWS"
    provider_disk_iops          = 240
    provider_instance_size_name = "M30"

    replication_specs {
        zone_name  = "Zone 1"
        num_shards = 1
        regions_config {
        region_name     = "EU_CENTRAL_1"
        electable_nodes = 3
        priority        = 7
        read_only_nodes = 0
        }
    }

    replication_specs {
        zone_name  = "Zone 2"
        num_shards = 1
        regions_config {
        region_name     = "US_EAST_2"
        electable_nodes = 3
        priority        = 7
        read_only_nodes = 0
        }
```

```
        }
    }

    resource "mongodbatlas_global_cluster_config" "config" {
        project_id = mongodbatlas_cluster.test.project_id
        cluster_name = mongodbatlas_cluster.test.name

        managed_namespaces {
            db               = "mydata"
            collection       = "publishers"
            custom_shard_key = "city"
        }

        custom_zone_mappings {
            location ="CA"
            zone =  "Zone 1"
        }
    }
```

» Example Global cluster config

```
resource "mongodbatlas_cluster" "cluster-test" {
  project_id   = "<YOUR-PROJECT-ID>"
  name         = "cluster-test"
  num_shards   = 1

  replication_factor           = 3
  backup_enabled               = true
  auto_scaling_disk_gb_enabled = true
  mongo_db_major_version       = "4.0"

  //Provider Settings "block"
  provider_name               = "AWS"
  disk_size_gb                = 100
  provider_disk_iops          = 300
  provider_encrypt_ebs_volume = false
  provider_instance_size_name = "M40"
  provider_region_name        = "US_EAST_1"
}

resource "mongodbatlas_global_cluster_config" "config" {
    project_id   = mongodbatlas_cluster.test.project_id
    cluster_name = mongodbatlas_cluster.test.name

    managed_namespaces {
```

```
        db               = "mydata"
        collection       = "publishers"
        custom_shard_key = "city"
    }

    custom_zone_mappings {
        location = "CA"
        zone     = "Zone 1"
    }
}
```

» Argument Reference

- `project_id` - (Required) The unique ID for the project to create the database user.
- ‘cluster_name - (Required) The name of the Global Cluster.
- `managed_namespaces` - (Optional) Add a managed namespaces to a Global Cluster. For more information about managed namespaces, see Global Clusters. See Managed Namespace below for more details.
- `custom_zone_mappings` - (Optional) Each element in the list maps one ISO location code to a zone in your Global Cluster. See Custom Zone Mapping below for more details.

» Managed Namespace

- `collection` - (Required) The name of the collection associated with the managed namespace.
- `custom_shard_key` - (Required) The custom shard key for the collection. Global Clusters require a compound shard key consisting of a location field and a user-selected second key, the custom shard key.
- `db` - (Required) The name of the database containing the collection.

» Custom Zone Mapping

- `location` - (Required) The ISO location code to which you want to map a zone in your Global Cluster. You can find a list of all supported location codes here.
- `zone` - (Required) The name of the zone in your Global Cluster that you want to map to location.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The Terraform's unique identifier used internally for state management.
- `custom_zone_mapping` - A map of all custom zone mappings defined for the Global Cluster. Atlas automatically maps each location code to the closest geographical zone. Custom zone mappings allow administrators to override these automatic mappings. If your Global Cluster does not have any custom zone mappings, this document is empty.

» Import

Database users can be imported using project ID and cluster name, in the format `PROJECTID-CLUSTER_NAME`, e.g.

```
$ terraform import mongodbatlas_global_cluster_config.config 1112222b3bf99403840e8934-my-clu
```

See detailed information for arguments and attributes: MongoDB API Global Clusters

» mongodbatlas_alert_configuration

`mongodbatlas_alert_configuration` provides an Alert Configuration resource to define the conditions that trigger an alert and the methods of notification within a MongoDB Atlas project.

NOTE: Groups and projects are synonymous terms. You may find `groupId` in the official documentation.

» Example Usage

```
resource "mongodbatlas_alert_configuration" "test" {
  project_id = "<PROJECT-ID>"
  event_type = "OUTSIDE_METRIC_THRESHOLD"
  enabled    = true

  notification {
    type_name     = "GROUP"
    interval_min  = 5
    delay_min     = 0
    sms_enabled   = false
    email_enabled = true
  }

  matcher {
    field_name = "HOSTNAME_AND_PORT"
```

```
    operator   = "EQUALS"
    value      = "SECONDARY"
  }

  metric_threshold = {
    metric_name = "ASSERT_REGULAR"
    operator    = "LESS_THAN"
    threshold   = 99.0
    units       = "RAW"
    mode        = "AVERAGE"
  }
}
```

» Argument Reference

- `project_id` - (Required) The ID of the project where the alert configuration will create.
- `enabled` - It is not required, but If the attribute is omitted, by default will be false, and the configuration would be disabled. You must set true to enable the configuration.
- `event_type` - (Required) The type of event that will trigger an alert. Alert type Possible values:
  - Host
    - `OUTSIDE_METRIC_THRESHOLD`
    - `HOST_RESTARTED`
    - `HOST_UPGRADED`
    - `HOST_NOW_SECONDARY`
    - `HOST_NOW_PRIMARY`
  - Replica set
    - `NO_PRIMARY`
    - `TOO_MANY_ELECTIONS`
  - Sharded cluster
    - `CLUSTER_MONGOS_IS_MISSING`
    - `User`
    - `JOINED_GROUP`
    - `REMOVED_FROM_GROUP`
    - `USER_ROLES_CHANGED_AUDIT`
  - Project

    * USERS_AWAITING_APPROVAL
    * USERS_WITHOUT_MULTI_FACTOR_AUTH
    * GROUP_CREATED
  - Team

    * JOINED_TEAM
    * REMOVED_FROM_TEAM
  - Organization

    * INVITED_TO_ORG
    * JOINED_ORG
  - Data Explorer

    * DATA_EXPLORER
    * DATA_EXPLORER_CRUD
  - Billing

    * CREDIT_CARD_ABOUT_TO_EXPIRE
    * CHARGE_SUCCEEDED
    * INVOICE_CLOSED

  NOTE: If this is set to OUTSIDE_METRIC_THRESHOLD, the metricThreshold field must also be set.

» Matchers

Rules to apply when matching an object against this alert configuration. Only entities that match all these rules are checked for an alert condition. You can filter using the matchers array only when the eventTypeName specifies an event for a host, replica set, or sharded cluster.

- `field_name` - Name of the field in the target object to match on. Host alerts support these fields:
  - TYPE_NAME
  - HOSTNAME
  - PORT
  - HOSTNAME_AND_PORT
  - REPLICA_SET_NAME Replica set alerts support these fields:
  - REPLICA_SET_NAME
  - SHARD_NAME
  - CLUSTER_NAME Sharded cluster alerts support these fields:
  - CLUSTER_NAME
  - SHARD_NAME

All other types of alerts do not support matchers.

- `operator` - If omitted, the configuration is disabled.

- `value` - If omitted, the configuration is disabled.
- `operator` - The operator to test the field's value. Accepted values are:
  - `EQUALS`
  - `NOT_EQUALS`
  - `CONTAINS`
  - `NOT_CONTAINS`
  - `STARTS_WITH`
  - `ENDS_WITH`
  - `REGEX`
- `value` - Value to test with the specified operator. If `field_name` is set to TYPE_NAME, you can match on the following values:
  - `PRIMARY`
  - `SECONDARY`
  - `STANDALONE`
  - `CONFIG`
  - `MONGOS`

» Metric Threshold

The threshold that causes an alert to be triggered. Required if `event_type_name` : "OUTSIDE_METRIC_THRESHOLD".

- `metric_name` - Name of the metric to check.
- `operator` - Operator to apply when checking the current metric value against the threshold value. Accepted values are:
  - `GREATER_THAN`
  - `LESS_THAN`
- `threshold` - Threshold value outside of which an alert will be triggered.
- `units` - The units for the threshold value. Depends on the type of metric. Accepted values are:
  - `RAW`
  - `BITS`
  - `BYTES`
  - `KILOBITS`
  - `KILOBYTES`
  - `MEGABITS`
  - `MEGABYTES`
  - `GIGABITS`
  - `GIGABYTES`
  - `TERABYTES`
  - `PETABYTES`

  - MILLISECONDS
  - SECONDS
  - MINUTES
  - HOURS
  - DAYS

- `mode` - This must be set to AVERAGE. Atlas computes the current metric value as an average.

» Notifications

Notifications to send when an alert condition is detected.

- `api_token` - Slack API token. Required for the SLACK notifications type. If the token later becomes invalid, Atlas sends an email to the project owner and eventually removes the token.
- `channel_name` - Slack channel name. Required for the SLACK notifications type.
- `datadog_api_key` - Datadog API Key. Found in the Datadog dashboard. Required for the DATADOG notifications type.
- `datadog_region` - Region that indicates which API URL to use. Accepted regions are: `US`, `EU`. The default Datadog region is US.
- `delay_min` - Number of minutes to wait after an alert condition is detected before sending out the first notification.
- `email_address` - Email address to which alert notifications are sent. Required for the EMAIL notifications type.
- `email_enabled` - Flag indicating if email notifications should be sent. Configurable for `ORG`, `GROUP`, and `USER` notifications types.
- `flowdock_api_token` - The Flowdock personal API token. Required for the `FLOWDOCK` notifications type. If the token later becomes invalid, Atlas sends an email to the project owner and eventually removes the token.
- `flow_name` - Flowdock flow name in lower-case letters. Required for the `FLOWDOCK` notifications type
- `interval_min` - Number of minutes to wait between successive notifications for unacknowledged alerts that are not resolved. The minimum value is 5.
- `mobile_number` - Mobile number to which alert notifications are sent. Required for the SMS notifications type.

- `ops_genie_api_key` - Opsgenie API Key. Required for the `OPS_GENIE` notifications type. If the key later becomes invalid, Atlas sends an email to the project owner and eventually removes the token.
- `ops_genie_region` - Region that indicates which API URL to use. Accepted regions are: `US` ,`EU`. The default Opsgenie region is US.
- `org_name` - Flowdock organization name in lower-case letters. This is the name that appears after www.flowdock.com/app/ in the URL string. Required for the FLOWDOCK notifications type.
- `service_key` - PagerDuty service key. Required for the PAGER_DUTY notifications type. If the key later becomes invalid, Atlas sends an email to the project owner and eventually removes the key.
- `sms_enabled` - Flag indicating if text message notifications should be sent. Configurable for `ORG`, `GROUP`, and `USER` notifications types.
- `team_id` - Unique identifier of a team.
- `type_name` - Type of alert notification. Accepted values are:
  - `DATADOG`
  - `EMAIL`
  - `FLOWDOCK`
  - `GROUP` (Project)
  - `OPS_GENIE`
  - `ORG`
  - `PAGER_DUTY`
  - `SLACK`
  - `SMS`
  - `TEAM`
  - `USER`
  - `VICTOR_OPS`
  - `WEBHOOK`
- `username` - Name of the Atlas user to which to send notifications. Only a user in the project that owns the alert configuration is allowed here. Required for the `USER` notifications type.
- `victor_ops_api_key` - VictorOps API key. Required for the `VICTOR_OPS` notifications type. If the key later becomes invalid, Atlas sends an email to the project owner and eventually removes the key.
- `victor_ops_routing_key` - VictorOps routing key. Optional for the `VICTOR_OPS` notifications type. If the key later becomes invalid, Atlas sends an email to the project owner and eventually removes the key.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - Unique identifier used for terraform for internal manages and can be used to import.
- `alert_configuration_id` - Unique identifier for the alert configuration.
- `group_id` - Unique identifier of the project that owns this alert configuration.
- `created` - Timestamp in ISO 8601 date and time format in UTC when this alert configuration was created.
- `updated` - Timestamp in ISO 8601 date and time format in UTC when this alert configuration was last updated.

» Import

Alert Configuration can be imported using the `project_id-alert_configuration_id`, e.g.

```
$ terraform import mongodbatlas_alert_configuration.test 5d0f1f74cf09a29120e123cd-5d0f1f74c
```

For more information see: MongoDB Atlas API Reference.

» mongodbatlas_private_endpoint

`mongodbatlas_private_endpoint` provides a Private Endpoint resource. This represents a Private Endpoint Connection that can be created in an Atlas project.

IMPORTANT: You must have one of the following roles to successfully handle the resource: * Organization Owner * Project Owner

NOTE: Groups and projects are synonymous terms. You may find group_id in the official documentation.

» Example Usage

```
resource "mongodbatlas_private_endpoint" "test" {
  project_id    = "<PROJECT-ID>"
  provider_name = "AWS"
  region        = "us-east-1"
}
```

» Argument Reference

- `project_id` - Required Unique identifier for the project.
- `providerName` - (Required) Name of the cloud provider you want to create the private endpoint connection for. Must be AWS.
- `region` - (Required) Cloud provider region in which you want to create the private endpoint connection. Accepted values are:
  - `us-east-1`
  - `us-east-2`
  - `us-west-1`
  - `us-west-2`
  - `ca-central-1`
  - `sa-east-1`
  - `eu-north-1`
  - `eu-west-1`
  - `eu-west-2`
  - `eu-west-3`
  - `eu-central-1`
  - `me-south-1`
  - `ap-northeast-1`
  - `ap-northeast-2`
  - `ap-south-1`
  - `ap-southeast-1`
  - `ap-southeast-2`
  - `ap-east-1`

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The Terraform's unique identifier used internally for state management.
- `private_link_id` - Unique identifier of the AWS PrivateLink connection.
- `endpoint_service_name` - Name of the PrivateLink endpoint service in AWS. Returns null while the endpoint service is being created.
- `error_message` - Error message pertaining to the AWS PrivateLink connection. Returns null if there are no errors.
- `interface_endpoints` - Unique identifiers of the interface endpoints in your VPC that you added to the AWS PrivateLink connection.
- `status` - Status of the AWS PrivateLink connection. Returns one of the following values:
  - `INITIATING` Atlas is creating the network load balancer and VPC endpoint service.
  - `WAITING_FOR_USER` The Atlas network load balancer and VPC endpoint service are created and ready to receive connection requests.

When you receive this status, create an interface endpoint to continue configuring the AWS PrivateLink connection.
- `FAILED` A system failure has occurred.
- `DELETING` The AWS PrivateLink connection is being deleted.

» Import

Private Endpoint Connection can be imported using project ID and username, in the format `{project_id}-{private_link_id}`, e.g.

```
$ terraform import mongodbatlas_private_endpoint.test 1112222b3bf99403840e8934-3242342343112
```

See detailed information for arguments and attributes: MongoDB API Private Endpoint Connection

» mongodbatlas_private_endpoint_link

`mongodbatlas_private_endpoint_link` provides a Private Endpoint Link resource. This represents a Private Endpoint Link, which adds one interface endpoint to a private endpoint connection in an Atlas project.

IMPORTANT: You must have one of the following roles to successfully handle the resource: * Organization Owner * Project Owner

NOTE: Groups and projects are synonymous terms. You may find group_id in the official documentation.

» Example Usage

```
resource "mongodbatlas_private_endpoint" "test" {
  project_id    = "<PROJECT_ID>"
  provider_name = "AWS"
  region        = "us-east-1"
}

resource "aws_vpc_endpoint" "ptfe_service" {
  vpc_id             = "vpc-7fc0a543"
  service_name       = "${mongodbatlas_private_endpoint.test.endpoint_service_name}"
  vpc_endpoint_type  = "Interface"
  subnet_ids         = ["subnet-de0406d2"]
  security_group_ids = ["sg-3f238186"]
}

resource "mongodbatlas_private_endpoint_link" "test" {
```

```
  project_id            = "${mongodbatlas_private_endpoint.test.project_id}"
  private_link_id       = "${mongodbatlas_private_endpoint.test.private_link_id}"
  interface_endpoint_id = "${aws_vpc_endpoint.ptfe_service.id}"
}
```

» Argument Reference

- `project_id` - (Required) Unique identifier for the project.
- `private_link_id` - (Required) Unique identifier of the AWS PrivateLink connection which is created by `mongodbatlas_private_endpoint` resource.
- `interface_endpoint_id` - (Required) Unique identifier of the interface endpoint you created in your VPC with the AWS resource.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `id` - The Terraform's unique identifier used internally for state management.
- `delete_requested` - Indicates if Atlas received a request to remove the interface endpoint from the private endpoint connection.
- `error_message` - Error message pertaining to the interface endpoint. Returns null if there are no errors.
- `connection_status` - Status of the interface endpoint. Returns one of the following values:
  - `NONE` - Atlas created the network load balancer and VPC endpoint service, but AWS hasn't yet created the VPC endpoint.
  - `PENDING_ACCEPTANCE` - AWS has received the connection request from your VPC endpoint to the Atlas VPC endpoint service.
  - `PENDING` - AWS is establishing the connection between your VPC endpoint and the Atlas VPC endpoint service.
  - `AVAILABLE` - Atlas VPC resources are connected to the VPC endpoint in your VPC. You can connect to Atlas clusters in this region using AWS PrivateLink.
  - `REJECTED` - AWS failed to establish a connection between Atlas VPC resources to the VPC endpoint in your VPC.
  - `DELETING` - Atlas is removing the interface endpoint from the private endpoint connection.

» Import

Private Endpoint Link Connection can be imported using project ID and username, in the format `{project_id}-{private_link_id}-{interface_endpoint_id}`,

e.g.

```
$ terraform import mongodbatlas_private_endpoint_link.test 1112222b3bf99403840e8934-32423423
```

See detailed information for arguments and attributes: MongoDB API Private Endpoint Link Connection

» mongodbatlas_x509_authentication_database_user

`mongodbatlas_x509_authentication_database_user` provides a X509 Authentication Database User resource. The mongodbatlas_x509_authentication_database_user resource lets you manage MongoDB users who authenticate using X.509 certificates. You can manage these X.509 certificates or let Atlas do it for you.

Management	Description
Atlas	Atlas manages your Certificate Authority and can generate certificates for your MongoDB use
Customer	You must provide a Certificate Authority and generate certificates for your MongoDB users.

NOTE: Groups and projects are synonymous terms. You may find group_id in the official documentation.

» Example Usages

» Example Usage: Generate an Atlas-managed X.509 certificate for a MongoDB user

```
resource "mongodbatlas_database_user" "user" {
  project_id    = "<PROJECT-ID>"
  username      = "myUsername"
  x509_type     = "MANAGED"
  database_name = "$external"

  roles {
    role_name     = "atlasAdmin"
    database_name = "admin"
  }

  labels {
    key   = "My Key"
    value = "My Value"
  }
}
```

```
resource "mongodbatlas_x509_authentication_database_user" "test" {
  project_id              = "${mongodbatlas_database_user.user.project_id}"
  username                = "${mongodbatlas_database_user.user.username}"
  months_until_expiration = 2
}
```

» Example Usage: Save a customer-managed X.509 configuration for an Atlas project

```
resource "mongodbatlas_x509_authentication_database_user" "test" {
  project_id        = "<PROJECT-ID>"
  customer_x509_cas = <<-EOT
    -----BEGIN CERTIFICATE-----
    MIICmTCCAgICCQDZnHzklxsT9TANBgkqhkiG9w0BAQsFADCBkDELMAkGA1UEBhMC
    VVMxDjAMBgNVBAgMBVRleGFzMQ8wDQYDVQQHDAZBdXN0aW4xETAPBgNVBAoMCHRl
    c3QuY29tMQ0wCwYDVQQLDARUZXN0MREwDwYDVQQDDAh0ZXN0LmNvbTErMCkGCSqG
    SIb3DQEJARYcbWVsaXNzYS5wbHVua2V0dEBtb25nb2RiLmNvbTAeFw0yMDAyMDQy
    MDQ2MDFaFw0yMTAyMDMyMDQ2MDFaMIGQMQswCQYDVQQGEwJVUzEOMAwGA1UECAwF
    VGV4YXMxDzANBgNVBAcMBkF1c3RpbjERMA8GA1UECgwIdGVzdC5jb20xDTALBgNV
    BAsMBFRlc3QxETAPBgNVBAMMCHRlc3QuY29tMSswKQYJKoZIhvcNAQkBFhxtZWxp
    c3NhLnBsdW5rZXR0QG1vbmdvZGIuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCB
    iQKBgQCf1LRqr1zftzdYx2Aj9G76tb0noMPtj6faGLlPji1+m6Rn7RWD9L0ntWAr
    cURxvypa9jZ9MXFzDtLevvd3tHEmfrUT3ukNDX6+Jtc4kWm+Dh2A7OPd+deKZ2/O
    Fh8audEKAESGXnTbeJCeQa1XKlIkjqQHBNwES5h1b9vJtFoLJwIDAQABMA0GCSqG
    SIb3DQEBCwUAA4GBADMUncjEPV/MiZUcVNGmktP6BPmEqMXQWUDpdGW2+Tg2JtUA
    7MMILtepBkFzLO+GlpZxeAlXO0wxiNgEmCRONgh4+t2w3e7a8GFijYQ99FHrAC5A
    iul59bdl18gVqXia1Yeq/iK7Ohfy/Jwd7Hsm530elwkM/ZEkYDjBlZSXYdyz
    -----END CERTIFICATE-----"
  EOT
}
```

» Argument Reference

- `project_id` - (Required) Identifier for the Atlas project associated with the X.509 configuration.
- `months_until_expiration` - (Required) A number of months that the created certificate is valid for before expiry, up to 24 months. By default is 3.
- `username` - (Optional) Username of the database user to create a certificate for.
- `customer_x509_cas` - (Optional) PEM string containing one or more customer CAs for database user authentication.

» Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `current_certificate` - Contains the last X.509 certificate and private key created for a database user.

Certificates * `certificates` - Array of objects where each details one unexpired database user certificate.

- `certificates.#.id` - Serial number of this certificate.
- `certificates.#.created_at` - Timestamp in ISO 8601 date and time format in UTC when Atlas created this X.509 certificate.
- `certificates.#.group_id` - Unique identifier of the Atlas project to which this certificate belongs.
- `certificates.#.not_after` - Timestamp in ISO 8601 date and time format in UTC when this certificate expires.
- `certificates.#.subject` - Fully distinguished name of the database user to which this certificate belongs. To learn more, see RFC 2253.

» Import

X.509 Certificates for a User can be imported using project ID and username, in the format `project_id-username`, e.g.

```
$ terraform import mongodbatlas_x509_authentication_database_user.test 1112222b3bf99403840e8
```

For more information see: MongoDB Atlas API Reference.

Current X.509 Configuration can be imported using project ID, in the format `project_id`, e.g.

```
$ terraform import mongodbatlas_x509_authentication_database_user.test 1112222b3bf99403840e8
```

For more information see: MongoDB Atlas API Reference.