MongoDB Atlas Provider

The MongoDB Atlas provider is used to interact with the resources supported by MongoDB Atlas Services. The provider needs to be configured with the proper credentials before it can be used.

Use the navigation to the left to read about the available resources.

Example Usage

```
# Configure the MongoDB Atlas Provider
provider "mongodbatlas" {
  public_key = "${var.mongodbatlas_public_key}"
  private_key  = "${var.mongodbatlas_private_key}"
}

#Create the resources
```

Authentication

The MongoDB Atlas provider offers a flexible means of providing credentials for authentication. The following methods are supported, in this order, and explained below:

Static credentials

Static credentials can be provided by adding the fowlloing attributes in-line in the MongoDB Atlas provider block:

Usage:

```
provider "mongodbatlas" {
  public_key = "" #required
  private_key  = "" #required
}
```

Environment variables

You can provide your credentials via environment variables, representing your MongoDB Atlas authentication.

```
provider "mongodbatlas" {}
```

Usage:

```
$ export MONGODB_ATLAS_PUBLIC_KEY="xxxx"
$ export MONGODB_ATLAS_PRIVATE_KEY="xxxx"
$ terraform plan
```

Argument Reference

In addition to generic `provider` arguments (https://www.terraform.io/docs/configuration/providers.html) (e.g. `alias` and `version`), the following arguments are supported in the MongoDB Atlas `provider` block:

- `public_key` - (Optional) This is the MongoDB Atlas API public_key. It must be provided, but it can also be sourced from the `MONGODB_ATLAS_PUBLIC_KEY` environment variable.
- `private_key` - (Optional) This is the MongoDB Atlas private_key. It must be provided, but it can also be sourced from the `MONGODB_ATLAS_PRIVATE_KEY` environment variable.

For more information about how to get this programmatic API Keys see the following link (https://docs.atlas.mongodb.com/configure-api-access/#manage-programmatic-access-to-an-organization).

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. (https://docs.atlas.mongodb.com/reference/api/cloud-provider-snapshot-get-one/)

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. (https://docs.atlas.mongodb.com/reference/api/cloud-provider-snapshot-restore-jobs-get-one/)

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. (https://docs.atlas.mongodb.com/reference/api/cloud-provider-snapshot-restore-jobs-get-all/)

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. (https://docs.atlas.mongodb.com/reference/api/cloud-provider-snapshot-get-all/)

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 (https://docs.atlas.mongodb.com/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.
- `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.
- `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.
- `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 (https://docs.atlas.mongodb.com/reference/amazon-aws/), GCP (https://docs.atlas.mongodb.com/reference/google-gcp/), Azure (https://docs.atlas.mongodb.com/reference/microsoft-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 (https://docs.mongodb.com/manual/core/read-preference/) and readPreferenceTags (https://docs.mongodb.com/manual/core/read-preference/#tag-sets) options. For details on BI Connector for Atlas read preferences, refer to the BI Connector Read Preferences Table (https://docs.atlas.mongodb.com/tutorial/create-global-writes-cluster/#bic-read-preferences).

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.

See detailed information for arguments and attributes: MongoDB API Clusters (https://docs.atlas.mongodb.com/reference/api/clusters-create-one/)

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 (https://docs.atlas.mongodb.com/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.
- `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.
- `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.
- `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 (https://docs.atlas.mongodb.com/reference/amazon-aws/), GCP (https://docs.atlas.mongodb.com/reference/google-gcp/), Azure (https://docs.atlas.mongodb.com/reference/microsoft-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 (https://docs.mongodb.com/manual/core/read-preference/) and readPreferenceTags (https://docs.mongodb.com/manual/core/read-preference/#tag-sets) options. For details on BI Connector for Atlas read preferences, refer to the BI Connector Read Preferences Table (https://docs.atlas.mongodb.com/tutorial/create-global-writes-cluster/#bic-read-preferences).

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.

See detailed information for arguments and attributes: MongoDB API Clusters (https://docs.atlas.mongodb.com/reference/api/clusters-create-one/)

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>"
    database_name = "admin"

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

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

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.

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.
- `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.

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).

See MongoDB Atlas API (https://docs.atlas.mongodb.com/reference/api/database-users-get-single-user/) 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>"
    database_name = "admin"

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

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

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.
- `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.

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).

See MongoDB Atlas API (https://docs.atlas.mongodb.com/reference/api/database-users-get-single-user/) Documentation for more information.

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 (https://tools.ietf.org/html/rfc1918.html#section-3).

- `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 (https://docs.atlas.mongodb.com/reference/api/vpc-create-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
}
```

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` - 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 (https://tools.ietf.org/html/rfc1918.html#section-3).
- `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 (https://docs.atlas.mongodb.com/reference/api/vpc-get-containers-list/)

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 (https://docs.atlas.mongodb.com/reference/amazon-aws/).
- `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 (https://docs.atlas.mongodb.com/reference/api/vpc-get-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 (https://docs.atlas.mongodb.com/reference/amazon-aws/).

- `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 (https://docs.atlas.mongodb.com/reference/api/vpc-get-connections-list/)

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-MyClusterTest-5d116
d82014b764445b2f9b5
```

For more information see: MongoDB Atlas API Reference. (https://docs.atlas.mongodb.com/reference/api/cloud-provider-snapshot/)

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 5cf5a45a9ccf6400e60981b6-MyClust
er-5d1b654ecf09a24b888f4c79
```

For more information see: MongoDB Atlas API Reference. (https://docs.atlas.mongodb.com/reference/api/cloud-provider-snapshot-restore-jobs/)

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:
• Changes to cluster configurations can affect costs. Before making changes, please see Billing (https://docs.atlas.mongodb.com/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
  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
  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
  backup_enabled          = false
  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
    }
  }
}
```

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 (https://docs.atlas.mongodb.com/reference/api/clusters-create-one/) `providerSettings.instanceSizeName` for valid values and default resources.
- `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 (https://docs.atlas.mongodb.com/reference/api/clusters-create-one/) `providerSettings.instanceSizeName` for valid values and default resources.
- `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.

  The default value is false.
- `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) 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, and 32GB for Azure. 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.

- `mongo_db_major_version` - (Optional) Version of the cluster to deploy. Atlas supports the following MongoDB versions for M10+ clusters: `3.4`, `3.6` or `4.0`. You must set this value to `4.0` 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.
- `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. (Note: When upgrading from a multi-tenant cluster to a dedicated cluster remove this argument.)

  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 disk type of the server's root volume. If omitted, Atlas uses the default disk type for the selected providerSettings.instanceSizeName.
- `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 (https://docs.atlas.mongodb.com/reference/amazon-aws/), GCP (https://docs.atlas.mongodb.com/reference/google-gcp/), Azure (https://docs.atlas.mongodb.com/reference/microsoft-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 (https://docs.mongodb.com/manual/core/read-preference/) and readPreferenceTags (https://docs.mongodb.com/manual/core/read-preference/#tag-sets) options. For details on BI Connector for Atlas read preferences, refer to the BI Connector Read Preferences Table (https://docs.atlas.mongodb.com/tutorial/create-global-writes-cluster/#bic-read-preferences).
    - 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.

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 (https://docs.atlas.mongodb.com/reference/api/clusters-create-one/)

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. (https://www.terraform.io/docs/state/sensitive-data.html)

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     = "%s"
        database_name = "admin"
    }
}
```

Argument Reference

- `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` - (Optional) User's initial password. This is required to create the user but may be removed after. Password may show up in logs, and it will be stored in the state file as plain-text. Password can be changed in the web interface to increase security.

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 (https://docs.atlas.mongodb.com/reference/api/database-users-create-a-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).

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 `PROJECTID-USERNAME`, e.g.

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

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

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/d891821e3d364e9eb88fbd
3d11807b86"
    secret              = "EXAMPLESECRET"
    tenant_id           = "e8e4b6ba-ff32-4c88-a9af-EXAMPLEID"
  }

  google_cloud_kms = {
    enabled                 = true
    service_account_key     = "{\"type\": \"service_account\",\"project_id\": \"my-project-common-0\",\"p
rivate_key_id\": \"e120598ea4f88249469fcdd75a9a785c1bb3\",\"private_key\": \"-----BEGIN PRIVATE KEY-----\
\nMIIEuwIBA(truncated)SfecnS0mT94D9\\n-----END PRIVATE KEY-----\\n\",\"client_email\": \"my-email-kms-0@m
y-project-common-0.iam.gserviceaccount.com\",\"client_id\": \"10180967717292066\",\"auth_uri\": \"https:/
/accounts.google.com/o/oauth2/auth\",\"token_uri\": \"https://accounts.google.com/o/oauth2/token\",\"auth
_provider_x509_cert_url\": \"https://www.googleapis.com/oauth2/v1/certs\",\"client_x509_cert_url\": \"htt
ps://www.googleapis.com/robot/v1/metadata/x509/my-email-kms-0%40my-project-common-0.iam.gserviceaccount.c
om\"}"
    key_version_resource_id = "projects/my-project-common-0/locations/us-east4/keyRings/my-key-ring-0/cry
ptoKeys/my-key-0/cryptoKeyVersions/1"
  }
}
```

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. (https://docs.atlas.mongodb.com/reference/api/encryption-at-rest/)

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.

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"
}

resource "mongodbatlas_network_peering" "test" {
  accepter_region_name    = "us-east-1"
  project_id              = mongodbatlas_network_container.test.project_id
  container_id            = mongodbatlas_network_container.test.container_id
  provider_name           = "AWS"
  route_table_cidr_block  = <AWS_VPC_CIDR_BLOCK>
  vpc_id                  = <AWS_VPC_ID>
  aws_account_id          = <AWS_ACCOUNT_ID>
}
```

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 (https://tools.ietf.org/html/rfc1918.html#section-3).
- `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 1112222b3bf99403840e8934-5cbf563d87d9d6725
3be590a
```

See detailed information for arguments and attributes: MongoDB API Network Peering Container (https://docs.atlas.mongodb.com/reference/api/vpc-create-container/)

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.

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

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>
  google_project_id = <your-google-project-id>
}
```

Example with AWS.

```
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"
}
```

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/${mongodbatlas_network_peering.test.atlas_gcp_project_id}/global/networks/${mo
ngodbatlas_network_peering.test.atlas_vpc_name}"
}
```

Example with Azure

```
resource "mongodbatlas_network_peering" "test" {
  project_id            = local.project_id
  atlas_cidr_block      = "192.168.0.0/21"
  container_id          = "507f1f77bcf86cd799439011"
  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"
}
```

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.

- `accepter_region_name` - (Optional | **AWS Required**) Specifies the region where the peer VPC resides. For complete lists of supported regions, see Amazon Web Services (https://docs.atlas.mongodb.com/reference/amazon-aws/).
- `aws_account_id` - (Optional | **AWS Required**) Account ID of the owner of the peer VPC.
- `provider_name` - (Optional) Cloud provider for this VPC peering connection. If omitted, Atlas sets this parameter to AWS. (Possible Values `AWS`, `AZURE`, `GCP`).
- `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 (https://docs.atlas.mongodb.com/reference/amazon-aws/).
- `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 1112222b3bf99403840e8934-5cbf563d87d9d67253be590a-AWS
```

See detailed information for arguments and attributes: MongoDB API Network Peering Connection (https://docs.atlas.mongodb.com/reference/api/vpc-create-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. (https://docs.atlas.mongodb.com/reference/api/get-private-ip-mode-for-project/)

mongodbatlas_project

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

Example Usage

```
resource "mongodbatlas_project" "my_project" {
    name   = "testacc-project"
    org_id = "5b93ff2f96e82120w0aaec19"
}
```

Argument Reference

- `name` - (Required) The name of the project you want to create.
- `org_id` - (Required) The ID of the organization you want to create the project within.

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. (https://docs.atlas.mongodb.com/reference/api/projects/)

mongodbatlas_project_ip_whitelist

`mongodbatlas_project_ip_whitelist` provides an IP Whitelist entry resource. The whitelist grants access from IPs or CIDRs to clusters within the Project.

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

Example Usage

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

    whitelist {
      cidr_block = "1.2.3.4/32"
      comment    = "cidr block for tf acc testing"
    }
    whitelist {
      ip_address = "2.3.4.5"
      comment    = "ip address for tf acc testing"
    }
    whitelist {
      cidr_block = "3.4.5.6/32"
      comment    = "cidr block for tf acc testing"
    }
    whitelist {
      ip_address = "4.5.6.7"
      comment    = "ip address for tf acc testing"
    }
 }
```

Argument Reference

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

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`, e.g.

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

For more information see: MongoDB Atlas API Reference. (https://docs.atlas.mongodb.com/reference/api/whitelist/)