Azure Provider

The Azure Provider can be used to configure infrastructure in Microsoft Azure using the Azure Resource Manager API's. Documentation regarding the Data Sources and Resources supported by the Azure Provider can be found in the navigation to the left.
To learn the basics of Terraform using this provider, follow the hands-on get started tutorials.
Interested in the provider's latest features, or want to make sure you're up to date? Check out the changelog for version information and release notes.

Authenticating to Azure

Terraform supports a number of different methods for authenticating to Azure:

- Authenticating to Azure using the Azure CLI
- Authenticating to Azure using Managed Service Identity
- Authenticating to Azure using a Service Principal and a Client Certificate
- Authenticating to Azure using a Service Principal and a Client Secret
- Authenticating to Azure using OpenID Connect

We recommend using either a Service Principal or Managed Service Identity when running Terraform non-interactively (such as when running Terraform in a CI server) - and authenticating using the Azure CLI when running Terraform locally.

Note:

The User, Service Principal or Managed Identity running Terraform should have permissions to register Azure Resource Providers. If the principal running Terraform has insufficient permissions to register Resource Providers then we recommend setting the property skip_provider_registration in the provider block to prevent auto-registration.

Example Usage

```
# We strongly recommend using the required_providers block to set the
# Azure Provider source and version being used
terraform {
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
```

```
      version = "=3.0.0"
    }
  }
}

# Configure the Microsoft Azure Provider
provider "azurerm" {
  skip_provider_registration = true # This is only required when the User, Service Principal, or Identity running Terraform lacks the permissions to register Azure Resource Providers.
  features {}
}

# Create a resource group
resource "azurerm_resource_group" "example" {
  name     = "example-resources"
  location = "West Europe"
}

# Create a virtual network within the resource group
resource "azurerm_virtual_network" "example" {
  name                = "example-network"
  resource_group_name = azurerm_resource_group.example.name
  location            = azurerm_resource_group.example.location
  address_space       = ["10.0.0.0/16"]
}
```

Bugs and Feature Requests

The Azure provider's bugs and feature requests can be found in the GitHub repo issues. Please avoid "me too" or "+1" comments. Instead, use a thumbs up reaction on enhancement requests. Provider maintainers will often prioritize work based on the number of thumbs on an issue.
Community input is appreciated on outstanding issues! We love to hear what use cases you have for new features, and want to provide the best possible experience for you using the Azure provider.
If you have a bug or feature request without an existing issue

- if an existing resource or field is working in an unexpected way, file a bug.
- if you'd like the provider to support a new resource or field, file an enhancement/feature request.

The provider maintainers will often use the assignee field on an issue to mark who is working on it.

- An issue assigned to an individual maintainer indicates that the maintainer is working on the issue
- If you're interested in working on an issue please leave a comment on that issue

If you have configuration questions, or general questions about using the provider, try checking out:

- Terraform's community resources
- HashiCorp support for Terraform Enterprise customers

Argument Reference

The following arguments are supported:

- `features` - (Required) A `features` block as defined below which can be used to customize the behaviour of certain Azure Provider resources.
- `client_id` - (Optional) The Client ID which should be used. This can also be sourced from the `ARM_CLIENT_ID` Environment Variable.
- `client_id_file_path` (Optional) The path to a file containing the Client ID which should be used. This can also be sourced from the `ARM_CLIENT_ID_FILE_PATH` Environment Variable.
- `environment` - (Optional) The Cloud Environment which should be used. Possible values are `public`, `usgovernment`, `german`, and `china`. Defaults to `public`. This can also be sourced from the `ARM_ENVIRONMENT` Environment Variable.
- `subscription_id` - (Optional) The Subscription ID which should be used. This can also be sourced from the `ARM_SUBSCRIPTION_ID` Environment Variable.
- `tenant_id` - (Optional) The Tenant ID which should be used. This can also be sourced from the `ARM_TENANT_ID` Environment Variable.
- `auxiliary_tenant_ids` - (Optional) List of auxiliary Tenant IDs required for multi-tenancy and cross-tenant scenarios. This can also be sourced from the `ARM_AUXILIARY_TENANT_IDS` Environment Variable.

When authenticating as a Service Principal using a Client Certificate, the following fields can be set:

- `client_certificate` - (Optional) A base64-encoded PKCS#12 bundle to be used as the client certificate for authentication. This can also be sourced from the `ARM_CLIENT_CERTIFICATE` environment variable.
- `client_certificate_password` - (Optional) The password associated with the Client Certificate. This can also be sourced from the `ARM_CLIENT_CERTIFICATE_PASSWORD` Environment Variable.

- `client_certificate_path` - (Optional) The path to the Client Certificate associated with the Service Principal which should be used. This can also be sourced from the `ARM_CLIENT_CERTIFICATE_PATH` Environment Variable.

More information on how to configure a Service Principal using a Client Certificate can be found in this guide.

When authenticating as a Service Principal using a Client Secret, the following fields can be set:

- `client_secret` - (Optional) The Client Secret which should be used. This can also be sourced from the `ARM_CLIENT_SECRET` Environment Variable.
- `client_secret_file_path` - (Optional) The path to a file containing the Client Secret which should be used. This can also be sourced from the `ARM_CLIENT_SECRET_FILE_PATH` Environment Variable.

More information on how to configure a Service Principal using a Client Secret can be found in this guide.

When authenticating as a Service Principal using Open ID Connect, the following fields can be set:

- `oidc_request_token` - (Optional) The bearer token for the request to the OIDC provider. This can also be sourced from the `ARM_OIDC_REQUEST_TOKEN` or `ACTIONS_ID_TOKEN_REQUEST_TOKEN` Environment Variables.
- `oidc_request_url` - (Optional) The URL for the OIDC provider from which to request an ID token. This can also be sourced from the `ARM_OIDC_REQUEST_URL` or `ACTIONS_ID_TOKEN_REQUEST_URL` Environment Variables.
- `oidc_token` - (Optional) The ID token when authenticating using OpenID Connect (OIDC). This can also be sourced from the `ARM_OIDC_TOKEN` environment Variable.
- `oidc_token_file_path` - (Optional) The path to a file containing an ID token when authenticating using OpenID Connect (OIDC). This can also be sourced from the `ARM_OIDC_TOKEN_FILE_PATH` environment Variable.
- `use_oidc` - (Optional) Should OIDC be used for Authentication? This can also be sourced from the `ARM_USE_OIDC` Environment Variable. Defaults to `false`.

More information on how to configure a Service Principal using OpenID Connect can be found in this guide.

When authenticating using Managed Identity, the following fields can be set:

- `msi_endpoint` - (Optional) The path to a custom endpoint for Managed Identity - in most circumstances, this should be detected automatically. This can also be sourced from the `ARM_MSI_ENDPOINT` Environment Variable.
- `use_msi` - (Optional) Should Managed Identity be used for Authentication? This can also be sourced from the `ARM_USE_MSI` Environment Variable. Defaults to `false`.

More information on how to configure a Service Principal using Managed Identity can be found in this guide.

For Azure CLI authentication, the following fields can be set:

- `use_cli` - (Optional) Should Azure CLI be used for authentication? This can also be sourced from the `ARM_USE_CLI` environment variable. Defaults to `true`.

For some advanced scenarios, such as where more granular permissions are necessary - the following properties can be set:

- `disable_terraform_partner_id` - (Optional) Disable sending the Terraform Partner ID if a custom `partner_id` isn't specified, which allows Microsoft to better understand the usage of Terraform. The Partner ID does not give HashiCorp any direct access to usage information. This can also be sourced from the `ARM_DISABLE_TERRAFORM_PARTNER_ID` environment variable. Defaults to `false`.
- `metadata_host` - (Optional) The Hostname of the Azure Metadata Service (for example `management.azure.com`), used to obtain the Cloud Environment when using a Custom Azure Environment. This can also be sourced from the `ARM_METADATA_HOSTNAME` Environment Variable.

Note:
`environment` must be set to the requested environment name in the list of available environments held in the `metadata_host`.

- `partner_id` - (Optional) A GUID/UUID registered with Microsoft to facilitate partner resource usage attribution. This can also be sourced from the `ARM_PARTNER_ID` Environment Variable. Supported formats are `<guid>` / `pid-<guid>` (GUIDs registered in Partner Center) and `pid-<guid>-partnercenter` (for published commercial marketplace Azure apps).
- `auxiliary_tenant_ids` - (Optional) Contains a list of (up to 3) other Tenant IDs used for cross-tenant and multi-tenancy scenarios with multiple AzureRM provider definitions. The list of `auxiliary_tenant_ids` in a given AzureRM provider definition

contains the other, remote Tenants and should not include its own `subscription_id` (or `ARM_SUBSCRIPTION_ID` Environment Variable).

- `skip_provider_registration` - (Optional) Should the AzureRM Provider skip registering the Resource Providers it supports? This can also be sourced from the `ARM_SKIP_PROVIDER_REGISTRATION` Environment Variable. Defaults to `false`.

Note

By default, Terraform will attempt to register any Resource Providers that it supports, even if they're not used in your configurations to be able to display more helpful error messages. If you're running in an environment with restricted permissions, or wish to manage Resource Provider Registration outside of Terraform you may wish to disable this flag; however, please note that the error messages returned from Azure may be confusing as a result (example: `API version 2019-01-01 was not found for Microsoft.Foo`).

- `storage_use_azuread` - (Optional) Should the AzureRM Provider use AzureAD to connect to the Storage Blob & Queue API's, rather than the SharedKey from the Storage Account? This can also be sourced from the `ARM_STORAGE_USE_AZUREAD` Environment Variable. Defaults to `false`.

Note:

This requires that the User/Service Principal being used has the associated `Storage` roles - which are added to new Contributor/Owner role-assignments, but **have not** been backported by Azure to existing role-assignments.

Note:

The Files & Table Storage API's do not support authenticating via AzureAD and will continue to use a SharedKey to access the API's.

- `use_msal` - (Optional) When `true`, and when using service principal authentication, the provider will obtain v2 authentication tokens from the Microsoft Identity Platform. Has no effect when authenticating via Managed Identity or the Azure CLI. Can also be set via the `ARM_USE_MSAL` or `ARM_USE_MSGRAPH` environment variables.

Note:

This will behaviour will be defaulted on in version 3.0 of the AzureRM (with no opt-out) due to the deprecation of Azure Active Directory Graph.

It's also possible to use multiple Provider blocks within a single Terraform configuration, for example, to work with resources across multiple Subscriptions - more information can be found in the documentation for Providers.

Features

The `features` block allows configuring the behaviour of the Azure Provider, more information can be found on the dedicated page for the `features` block.