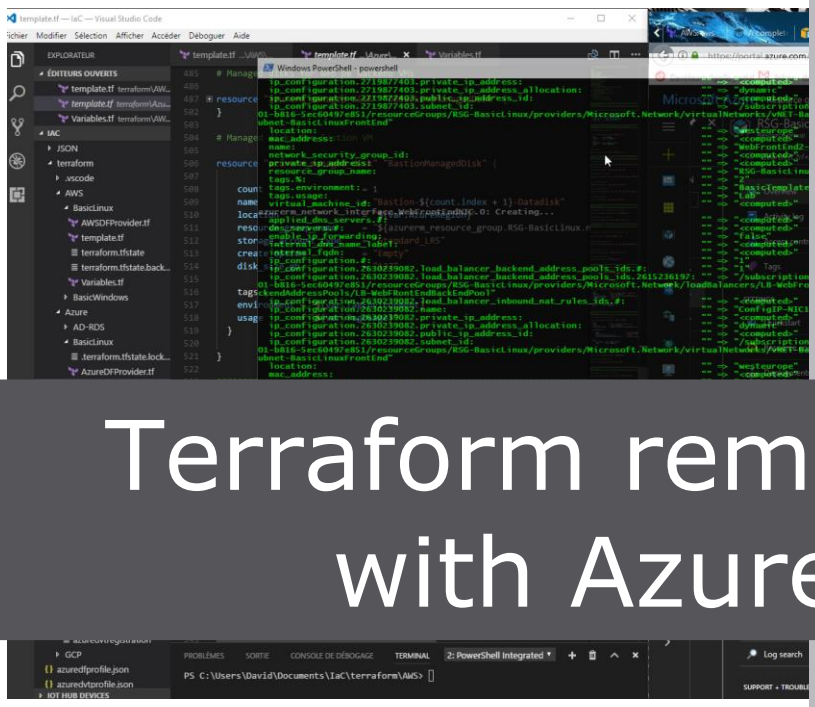


Teknews.cloud

Terraform remote state with Azure storage

David Frappart

28/05/2018

Contributor

Name	Title	Email
David Frappart	Cloud Architect	david@teknews.cloud

Version Control

Version	Date	State
1.0	2018/05/28	Final

Table of Contents

1 When you’re not working alone anymore with Terraform

I use terraform nearly every day and often i am plying alone in my tech guy cave. However, with the adoption of the public cloud paradigm, at one point, the playing alone concept should not be enough anymore. In this article, we will go back on what the terraform state is and how we can start working in team by using the remote state and Azure storage.

2 Terraform state reminder

2.1 About the terraform state file

When we use terraform to provision resource, a file named terraform.tfstate is created. It contains information about all the resource deployed through terraform. So without surprise it is a big file. Below is an extract of the file:


```
{
    "version": 3,
    "terraform_version": "0.11.7",
    "serial": 2,
    "lineage": "60237f35-ae6a-b190-56f8-04ece29b5999",
    "modules": [
        {
            "path": [
                "root"
            ],
            "outputs": {
                "DiagStorageAccountID": {
                    "sensitive": false,
                    "type": "string",
                    "value": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/RG-
PoCRemoteStateLab/providers/Microsoft.Storage/storageAccounts/stoaremotestatelablog
"
                },
                "DiagStorageAccountName": {
                    "sensitive": false,
                    "type": "string",
                    "value": "stoaremotestatelablog"
                },
```

```
                "DiagStorageAccountPrimaryAccessKey": {
                    "sensitive": true,
                    "type": "string",
                    "value":
"B1AjrhLzZ8WDHjgDXD5xHyy+0Ek618WqBDoYtMpATwddEelOTc6yvvgrxHRh94tuYtOMyEDi1ri4BLskhH
u2Zg=="
                },
                "DiagStorageAccountPrimaryBlobEP": {
                    "sensitive": false,
                    "type": "string",
                    "value":
"https://<Storage_Account_Name>.blob.core.windows.net/"
                },
                "DiagStorageAccountPrimaryFileEP": {
                    "sensitive": false,
                    "type": "string",
                    "value":
"https://<Storage_Account_Name>.file.core.windows.net/"
                },
```

Two things should be noted about this file. The first one is that since it contains ALL the configuration informations, it may also contains sensitives datas such as passwords provided in input or storage account keys output at the end of the apply step. The second one is more centered on the team working aspect. When the state is local, since it references the deployed infrastructure, it is thus tyed to the local disk and does not ease the collaboration.

2.2 Terraform remote state concept

Terraform developpers address the team work issue with the remote state. Typically, when configuring the remote state, the file containing the state is moved to a remote location, as the name implies.
The infrastructure code on its own is hosted on repository such as GitHub. In this regard, another team member can access the repo and start coding modification on the infrastructure. But what happens when the state remains local to the first team member? The infrastructure may be altered or replicated in an inconsistent state. While with the remote state, others team members may access the current state of the configuration and either add or remove object while taking the current config into account. Ok, so far we answered the need for team working. Let’s now see about securing sensitive data.

2.3 Terraform remote state in Azure storage

Just to be clear, Azure storage is neither the only option to provide a remote state solution for terraform nor the only one answering part of the security concerns. That being said, let’s have a look at it.

By default, data stored in a storage account is encrypted at rest. Meaning that sensitive data is not visible in clear. An important nformation, that may decrease security worries, is that it is possible to encrypted either through an Azure managed key or through an independant key, provided on the client's side. Also, https traffic can be enforced to avoid clear data in flight.
So it would seams that the only place where sensitive data may reside is in the memory of the system used to execute the plan and apply steps.
That's nice but that's about all. If someone tracks somehow the data displayed in his console, sensitive data will still be displayed and vulnerable to leaks. Unfortunetaly, there is currently no way to hide those sensitives datas except, quoting terraform documentation, "on a resource by resource basis" if so documented. One example of this is with the output data. A parameter in the output object can be added,providing an additional measure of security. There are however limits in that the values are still stored in the state and available with the terraform output command. Also, output interpolated in other module may still be displayed since there is no tracking of the sensitivity parameters accross the modules. Ways to improve security may resides in Hashicorp product vault to store passwords or others secrets provided in input usually in the code. Hopefully, developpment around terraform and Azure keyvault integration may provides an answer for this.

2.4 Another aspect to address when using remote state – locking the state

Once the team work starts, team members should be cautious about one fact. If the state is accessible to more than one person, the deployment should not be launch conccurently on multiple workstation. One advantage of Azure storage as a backend for the remote state is that it provides native capability to lock the state and thus answer the need for caution exposed before. Since the state is lock once one team guy (or girl) is applying the configuration, there is no risk that another person may also execute an apply at the same time. Obviously, people should have the same level of configuration available and that should be taken care of with proper planning for tracking code change.

3 Testing remote state on Azure storage

3.1 Configuring the backend

3.1.1 Backend Configuration details

The first step, before testing the remote test features discussed above, is to configure terraform to use a remote state instead of the local one. This step is completed by adding in the configuration the following code block:

```
terraform {
  backend "azurerm" {
```

```
    storage_account_name = "<Name_of_the_storage_account"
container_name           = "<Name_of_the_storage_container>"
    key                  = "remotestatetest.tf"
    access_key           = "<Key_allowing_access_to_the_storage_account>"
  }
}
```

3.1.2 Configuring Remote state when no previous plan was run

In the folowing example, no previous deployment was run. No current state exist. After running the init commmand, we get the following display:

```
PS D:\Users\User\Docs\IaC\terraform\Azure\RemoteStateTest> terraform init
Initializing modules...
- module.ResourceGroup
  Getting  source  "github.com/dfrappart/Terra-AZBasiclinuxWithModules//Modules//01
ResourceGroup/"
- module.SampleArchi_vNet
  Getting  source  "github.com/dfrappart/Terra-AZBasiclinuxWithModules//Modules//02
vNet/"
- module.DiagStorageAccount
  Getting  source  "github.com/dfrappart/Terra-AZBasiclinuxWithModules//Modules//03
StorageAccountGP/"
- module.LogStorageContainer
  Getting  source  "github.com/dfrappart/Terra-AZBasiclinuxWithModules//Modules//04
StorageAccountContainer/"
- module.FilesExchangeStorageAccount
  Getting  source  "github.com/dfrappart/Terra-AZBasiclinuxWithModules//Modules//03
StorageAccountGP/"
- module.InfraFileShare
  Getting  source  "github.com/dfrappart/Terra-AZBasiclinuxWithModules//Modules//05
StorageAccountShare"
- module.NSG_Subnet1
  Getting  source  "github.com/dfrappart/Terra-AZBasiclinuxWithModules//Modules//07
NSG/"
- module.Subnet1
  Getting  source  "github.com/dfrappart/Terra-AZBasiclinuxWithModules//Modules//06
Subnet/"
- module.AllowSSHFromInternetIn
  Getting  source  "github.com/dfrappart/Terra-AZBasiclinuxWithModules//Modules//08
NSGRule"
- module.AllowHTTPFromInternetVMIn
  Getting  source  "github.com/dfrappart/Terra-AZBasiclinuxWithModules//Modules//08
NSGRule"
```

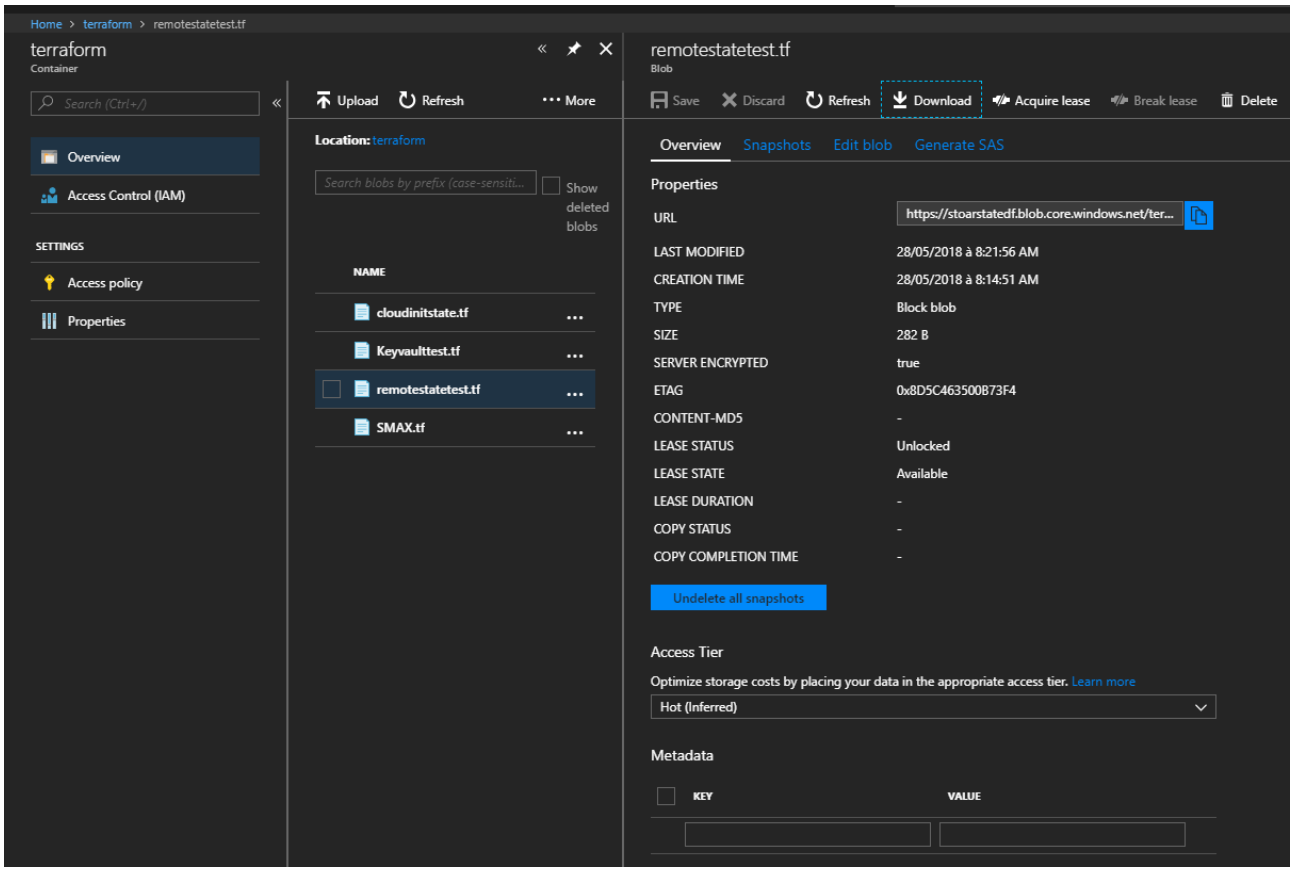
```
- module.VMPublicIP
  Getting  source  "github.com/dfrappart/Terra-AZBasiclinuxWithModules//Modules//10
PublicIP"
- module.AS_VM
  Getting  source  "github.com/dfrappart/Terra-AZBasiclinuxWithModules//Modules//13
AvailabilitySet"
- module.NICs_VM
  Getting  source  "github.com/dfrappart/Terra-AZBasiclinuxWithModules//Modules//12
NICwithPIPWithCount"
- module.DataDisks_VM
  Getting  source  "github.com/dfrappart/Terra-AZBasiclinuxWithModules//Modules//06
ManagedDiskswithcount"
- module.VMs_VM
  Getting        source        "github.com/dfrappart/Terra-AZCloudInit//Modules//01
LinuxVMWithCountwithCustomData"

Initializing the backend...

Successfully configured the backend "azurerm"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing provider plugins...
- Checking for available provider plugins on https://releases.hashicorp.com...
- Downloading plugin for provider "template" (1.0.0)...
- Downloading plugin for provider "azurerm" (1.6.0)...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.azurerm: version = "~> 1.6"
* provider.template: version = "~> 1.0"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
```

```
commands will detect it and remind you to do so if necessary.
```

At this point there is still no state stored on the Azure storage. Also, running the plan command will result only in a temporary file in the storage as display below:

```
PS D:\Users\User\Docs\IaC\terraform\Azure\RemoteStateTest> terraform plan
Acquiring state lock. This may take a few moments...
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.
```

After the plan command is completed, while the file specified to store the state remains, its content is empty:




```
{
    "version": 3,
    "serial": 1,
    "lineage": "cb3bed98-4e8d-8d73-ba56-4dc05d987a0b",
    "modules": [
        {
            "path": [
                "root"
```

```
            ],
            "outputs": {},
            "resources": {},
            "depends_on": []
        }
    ]
}
```

3.1.3 Configuring Remote state when a local state exists

In the following example, a local state already exist prior to run the plan command. Terraform detects the change of state configuration and thus display the following:

```
PS C:\Users\User\Documents\IaC\terraform\Azure\RemoteStateTest> terraform plan
Backend reinitialization required. Please run "terraform init".
Reason: Initial configuration of the requested backend "azurerm"

The "backend" is the interface that Terraform uses to store state,
perform operations, etc. If this message is showing up, it means that the
Terraform configuration you're using is using a custom configuration for
the Terraform backend.

Changes to backend configurations require reinitialization. This allows
Terraform to setup the new configuration, copy existing state, etc. This is
only done during "terraform init". Please run that command now then try again.

If the change reason above is incorrect, please verify your configuration
hasn't changed and try again. At this point, no changes to your existing
configuration or state have been made.

Failed to load backend: Initialization required. Please see the error message
above.
```

As the message says, a new terraform init is required.

```
PS C:\Users\User\Documents\IaC\terraform\Azure\RemoteStateTest> terraform init
Initializing modules...
- module.ResourceGroup
- module.SampleArchi_vNet
- module.DiagStorageAccount
- module.LogStorageContainer
- module.FilesExchangeStorageAccount
- module.InfraFileShare
```

```
- module.NSG_Subnet1
- module.Subnet1
- module.AllowSSHFromInternetIn
- module.AllowHTTPFromInternetVMIn
- module.VMPublicIP
- module.AS_VM
- module.NICs_VM
- module.DataDisks_VM
- module.VMs_VM

Initializing the backend...
Acquiring state lock. This may take a few moments...
Do you want to copy existing state to the new backend?
  Pre-existing state was found while migrating the previous "local" backend to the
  newly configured "azurerm" backend. An existing non-empty state already exists in
  the new backend. The two states have been saved to temporary files that will be
  removed after responding to this query.

  Previous (type "local"): C:\Users\User\AppData\Local\Temp\terraform095224915\1-
local.tfstate
  New      (type "azurerm"): C:\Users\User\AppData\Local\Temp\terraform095224915\2-
azurerm.tfstate

  Do you want to overwrite the state in the new backend with the previous state?
  Enter "yes" to copy and "no" to start with the existing state in the newly
  configured "azurerm" backend.

  Enter a value: yes

Releasing state lock. This may take a few moments...

Successfully configured the backend "azurerm"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing provider plugins...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.azurerm: version = "~> 1.3"
```

```
* provider.template: version = "~> 1.0"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to seeany
changes that are required for your infrastructure. All Terraform commands
should now work.
If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

After this command completes successsfully, the state is now stored in an Azure storage account.

3.2 Bad stuffs that may happen when configuring the backend

It should be obvious that network access to the Azure storage is required when initiating the remote state. As a matter of fact, i came accorss this error while working in a train thus with unreliable Internet access. Failure in the network connection give the following result:

```
PS C:\Users\User\Documents\IaC\terraform\Azure\RemoteStateTest> terraform init
Initializing modules...
- module.ResourceGroup
- module.SampleArchi_vNet
- module.DiagStorageAccount
- module.LogStorageContainer
- module.FilesExchangeStorageAccount
- module.InfraFileShare
- module.NSG_Subnet1
- module.Subnet1
- module.AllowSSHFromInternetIn
- module.AllowHTTPFromInternetVMIn
- module.VMPublicIP
- module.AS_VM
- module.NICs_VM
- module.DataDisks_VM
- module.VMs_VM

Initializing the backend...
Acquiring state lock. This may take a few moments...
Do you want to copy existing state to the new backend?
  Pre-existing state was found while migrating the previous "local" backend to the
  newly configured "azurerm" backend. An existing non-empty state already exists in
  the new backend. The two states have been saved to temporary files that will be
  removed after responding to this query.
```

```
  Previous  (type  "local"):  C:\Users\User\AppData\Local\Temp\terraform095224915\1-
local.tfstate
  New       (type "azurerm"): C:\Users\User\AppData\Local\Temp\terraform095224915\2-
azurerm.tfstate

  Do you want to overwrite the state in the new backend with the previous state?
  Enter "yes" to copy and "no" to start with the existing state in the newly
  configured "azurerm" backend.

  Enter a value: yes

Releasing state lock. This may take a few moments...

Error releasing the state lock!

Error      message:      failed      to      retrieve      lock      info:      Get
https://<BlobPath>.blob.core.windows.net/terraform/remotestatetest.tf?comp=metadata
: dial tcp xxx.xxx.xxx.xxx:443: connectex: Une tentative de connexion a échoué car
le parti connecté n’a pas répondu convenablement au-delà d’une certaine durée ou une
connexion établie a échoué car l’hôte de connexion n’a pas répondu.

Terraform acquires a lock when accessing your state to prevent others
running Terraform to potentially modify the state at the same time. An
error occurred while releasing this lock. This could mean that the lock
did or did not release properly. If the lock didn't release properly,
Terraform may not be able to run future commands since it'll appear as if
the lock is held.

In this scenario, please call the "force-unlock" command to unlock the
state manually. This is a very dangerous operation since if it is done
erroneously it could result in two people modifying state at the same time.
Only call this command if you're certain that the unlock above failed and
that no one else is holding a lock.

Error copying state from the previous "local" backend to the newly configured
"azurerm" backend:
    Head    https://stoarstatedf.blob.core.windows.net/terraform/remotestatetest.tf:
dial tcp 52.239.142.164:443: connectex: Une tentative de connexion a échoué car
le parti connecté n’a pas répondu convenablement au-delà d’une certaine durée ou une
connexion établie a échoué car l’hôte de connexion n’a pas répondu.

The state in the previous backend remains intact and unmodified. Please resolve
the error above and try again.
```

At this point, we have different possibilities. If this is a first deployment, losing access to the state is not an issue. We have the possibilitiy to either change the remote state config and run a new init or we can access the storage account to unlock the state file by breaking the lease on the blob object.
On the other hand, if this is not a first deployment, switching to another state file is not an option. So one should treats the issue carefully here to avoid the loss of the state and thus breaking the dependancy of the deployed resource and the terraform state. At worst, it may be necessary to import manually the resource in the state with the terraform import command. That is subject for another time however. In case of the Azure storage backend, the easiest way is to break the lease on the blob in the storage account.

3.3 Going back to local state

If the need ever arise to go back to local state, removing the code declaring the remote state is the first step. Then, as it is described in Terraform documentation, a new init step is required. Forgetting to do so will result in the following message:

```
PS C:\Users\User\Documents\IaC\terraform\Azure\RemoteStateTest> terraform plan
Backend reinitialization required. Please run "terraform init".
Reason: Unsetting the previously set backend "azurerm"

The "backend" is the interface that Terraform uses to store state,
perform operations, etc. If this message is showing up, it means that the
Terraform configuration you're using is using a custom configuration for
the Terraform backend.

Changes to backend configurations require reinitialization. This allows
Terraform to setup the new configuration, copy existing state, etc. This is
only done during "terraform init". Please run that command now then try again.

If the change reason above is incorrect, please verify your configuration
hasn't changed and try again. At this point, no changes to your existing
configuration or state have been made.

Failed to load backend: Initialization required. Please see the error message
above.
```

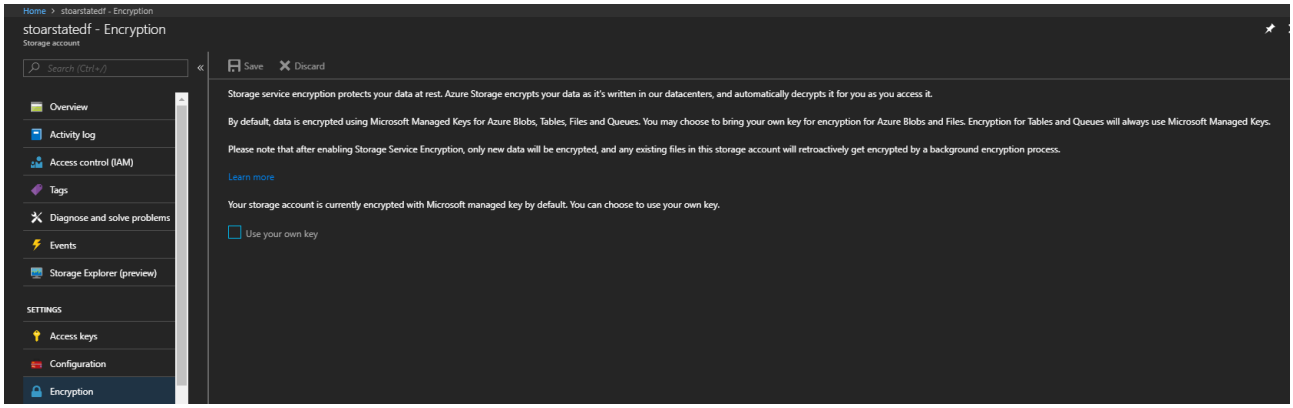
After a proper init, the following is displayed and we are back to a local state:

```
Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.
```

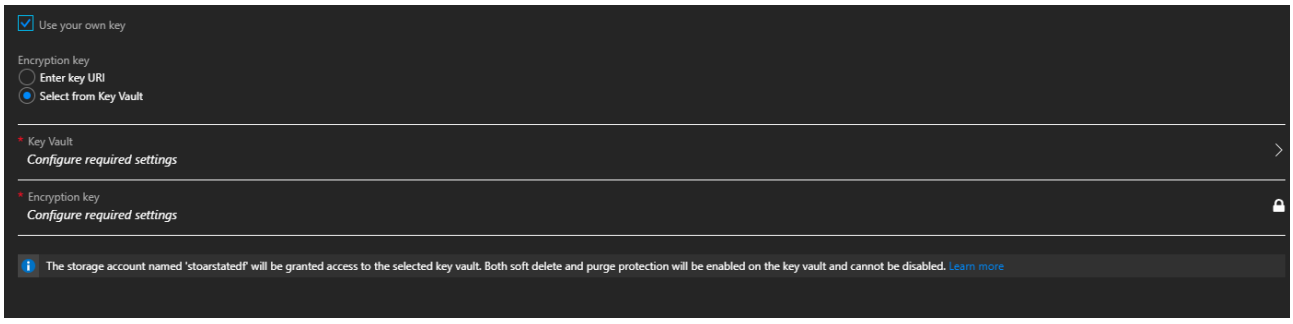
```
If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

3.4 Checking the security features on Azure storage

Azure storage account is provided with encryption at rest. By default, the key used for encryption is Microsoft managed. However, an option to bring client-managed key is available as displayed below



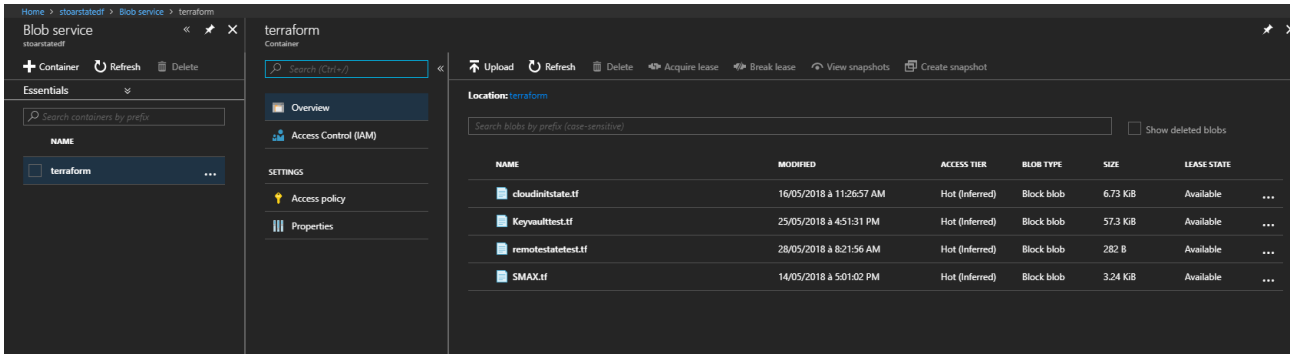
When the option is checked, an uri where the key is availableis required. This uri can be the uri of an Azure KeyVault:



3.5 Testing the lock state capability of Azure storage

As discussed earlier, each object stored in the account is sibject to a lease. When the object is leased, it cannot be tampered until the lock on the lease is released.

In the case of two IT people working at the same time on the configuration, it avoids tthat more than one apply happens at the same time:



When trying to run an Apply while another is already ongoing, the following is displayed:

```
PS D:\Users\User\Docs\IaC\terraform\Azure\RemoteStateTest> terraform apply
Acquiring state lock. This may take a few moments...

Error: Error locking state: Error acquiring the state lock: storage: service returned
error: StatusCode=409, ErrorCode=LeaseAlreadyPresent, ErrorMessage=There is already
a lease present.
RequestId:0adafa42-f01e-009a-2a5b-f607c9000000
Time:2018-05-28T08:13:50.1627969Z, RequestInitiated=Mon, 28 May 2018 08:13:49 GMT,
RequestId=0adafa42-f01e-009a-2a5b-f607c9000000,       API       Version=2016-05-31,
QueryParameterName=, QueryParameterValue=
Lock Info:
  ID:        d1c213fc-df99-f0a8-2db7-8fe6039bd73f
  Path:      terraform/remotestatetest.tf
  Operation: OperationTypeApply
  Who:       DF-LAPTOP\David@DF-Laptop
  Version:   0.11.7
  Created:   2018-05-28 08:13:19.0033342 +0000 UTC
  Info:


Terraform acquires a state lock to protect the state from being written
by multiple users at the same time. Please resolve the issue above and try
again. For most commands, you can disable locking with the "-lock=false"
flag, but this is not recommended.
```

Thus we can see that Azure storage as a backend provides natively a lock capability for the state.

4 Conclusion

In this article, we looked into the remote state options with Azure storage and had a look at how it can secure the deployment, in terms of sotring sensitive data but also in terms of securing deployment.
Azure storage being encrypted at rest, the information contained in the state aren't visible unless with the proper credentials in Azure Active Directory. Also, deployement is more team ready once this feature is configured.
Terraform offers even more capabilities for working in team and so daes Azure. But that will be for another time.

Another Tech blog

My thoughts and experiences on Cloud related tek