# Terraform automation

---

## How to handle existing AWS infrastructure

In order to bring any existing infrastructure under the control of Terraform two basic things need to happen:

1. Terraform's state file needs to reflect the existing environment. This can be done via two methods:

   a. The resources are imported using a "terraform import" command.

   b. If appropriate Terraform data resources need ot be synchronized using a "terraform refresh" command.

2. If method 1a is used we need to construct also a fully formed <name>.tf file(s) to represent the resource(s).

## aws2tf

There are tools that save us from the tedium of figuring out the correct "terraform import" commands and how to write a representative terraform file. And thus can bring an existing infrastructure under control of Terraform.

One such tool is "aws2tf"

Check if the directory aws2tf exists:

```
ls ~/environment/aws2tf
```

If it does not run:

```
cd ~/environment
git clone https://github.com/aws-samples/aws2tf.git
```

Now we are going to use aws2tf to:

- Import the terraform state from existing AWS resources.
- Generate the terraform files for the configuration.
- Perform a "terraform plan" to validate the configuration files generated.

**The following operation takes 10-15 minutes**

```
cd ~/environment/aws2tf
./aws2tf.sh
```

Lots of output is produced as aws2tf:

- Loops through each AWS resource type
- Performs the necessary `terraform import` commands
- Creates the requited *.tf configuration files in the "generated" directory
- And finally runs a 'terraform plan'

You may notice a few errors like:

```
Found Error: Error: Attribute name required exiting .... (pass for now)
```

2. Introduction to Kubernetes
3. Start the Workshop
4. Terraform Primer (Optional)
  1. Building a VPC with Terraform
    Terraform Networking - Part 1
    Terraform Networking - Part 2
    Terraform Networking - Part 3
  2. Validating Connectivity
    Testing - Part 1
  3. Creating a Second VPC - with one character !
    Second VPC - with one Character !
  Optional - Automatic generation of Terraform
    **Terraform automation**
  4. Cleanup
    Cleanup
5. Creating a private EKS

They can be ignored, after a few minutes you should see:

```
------------------------------------------------------------------------

No changes. Infrastructure is up-to-date.

This means that Terraform did not detect any differences between your
configuration and real physical resources that exist. As a result, no
actions need to be performed.
---------------------------------------------------------------------------
aws2tf output files are in generated/tf.903233334942
---------------------------------------------------------------------------
```

Change to this directory and observe all the Terraform files that have been created for you:

```
cd ~/environment/aws2tf/generated/tf.*
ls
```

Next issue a terraform plan and check no changes will be made - and thus the automatically generated Terraform files and imported state are valid:

```
terraform plan
```

---

In your own environment you could now switch AWS Accounts eg. from Development to Test issue the terraform plan and apply commands and you'd get an exact replica of the development environment in test - all without writing any code!

no "count" variables, format statements etc. or other great Terraform techniques such as using Terraform Modules.

---

Previous Next