

AWS, Terraform and Ansible lab

March 2018

AWS DEvops hands on lab

# Getting started

## Download Lab materials

Visit this URL to get the lab materials

<https://s3.eu-west-2.amazonaws.com/aws-demo-day/slalom-aws-lab/slalom-aws-lab.zip>

Unzip the materials using the password provided. It should form the following folder structure/contents:

./aws-demo-files

/ssh keys

## Connect to the build server

Within the lab materials there is a folder called ssh keys. It contains SSH keys which will be used to authenticate and connect to the ‘build’ server. Use ssh (mac/linux) or putty (MS Windows) connect to the build server as outlined below.

### Connect via ssh (MacOS/Linux)

Navigate to the folder that has the ssh keys and run the command below to set proper permission on the ssh key

chmod 400 slalom-demo-lon.pem

Run the following command to connect to the ‘build’ server.

ssh -i "slalom-demo-lon.pem" ec2-user@ec2-52-56-217-199.eu-west-2.compute.amazonaws.com

You should now be at the EC2 command prompt for the build server.

### Connect via Putty (Windows)

For MS Windows users Putty can be downloaded from here

<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

Once installed open Putty and in the menu on the left navigate to:

***Connection – SSH – Auth***

Click the browse button and navigate to the lab materials folder and locate the private key file:

### slalom-aws-lab\aws-demo-files\ssh keys\slalom-demo-lon.ppk

From the menu on the left navigate to ***Connection*** and add the line below the **Host Name** field, replace the <number> with your assigned labuser number e.g labuser12

labuser<number>@ec2-35-178-27-149.eu-west-2.compute.amazonaws.com

Click the Open button, if prompted to add the ssh key click OK. You should now be at the command prompt on the build server.

## Download the Lab Source code

From the build server run the following command to download the source code

git clone https://github.com/nmarchini/tf-01.git

Change directory to tf-01

cd tf-01

## Set Default AWS CLI Region and Output Format

The AWS region needs to be set using the AWS CLI *configure* command, enter the command below to set the region to eu-west-2.

aws configure [Enter]

AWS Access Key ID [None]: [Enter]

AWS Secret Access Key [None]: [Enter]

Default region name [None]: eu-west-2

Default output format [None]: json

# Getting ready to run terraform

## Setup the Terraform Backend

An S3 bucket and DynamoDB table need to be created to store the remote state and lock table.

### Create S3 Bucket

The code below creates a private S3 bucket, you need to edit it before you run the code, replace the <number> with your assigned labuser number. E.g labuser12

aws s3api create-bucket --acl private --bucket slalom0318-labuser<number> --region eu-west-2 --create-bucket-configuration LocationConstraint=eu-west-2

The expected output should look similar to this

"Location": http://slalom0318-labuser12.s3.amazonaws.com/

### Create DynamoDB table

The code below creates a DynamoDB table, you need to edit it before you run the code, replace the <number> with your assigned labuser number. E.g labuser12.

aws dynamodb create-table --table-name slalom0318-labuser<number> --attribute-definitions AttributeName=LockID,AttributeType=S --key-schema AttributeName=LockID,KeyType=HASH --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5

The expected output should look similar to this

"TableDescription": {

"TableArn": "arn:aws:dynamodb:eu-west-2:792599154768:table/slalom0318-labuser12",

"AttributeDefinitions": [

{

"AttributeName": "LockID",

"AttributeType": "S"

}

],

"ProvisionedThroughput": {

"NumberOfDecreasesToday": 0,

"WriteCapacityUnits": 5,

"ReadCapacityUnits": 5

},

"TableSizeBytes": 0,

"TableName": " slalom0318-labuser12",

"TableStatus": "CREATING",

"KeySchema": [

{

"KeyType": "HASH",

"AttributeName": "LockID"

}

],

"ItemCount": 0,

"CreationDateTime": 1520595158.146

}

# Running terraform

## State and workspace initialization

To initialise the backend in S3 run the command below, replace the <number> with your assigned labuser number. E.g labuser12.

STATE=vpc WORKSPACE=slalom0318-labuser<number> STATEBUCKET=slalom0318-labuser<number> STATELOCKTABLE=slalom0318-labuser<number> STATEREGION=eu-west-2 make first-run

This will initialize the remote state file in S3. The ouput should contain the following

Successfully configured the backend "s3"! Terraform will automatically use this backend unless the backend configuration changes.

Terraform has been successfully initialized!

Next the Terraform workspace needs to be initialized, replace the <number> with your assigned labuser number. E.g labuser12.

STATE=vpc WORKSPACE=slalom0318-labuser<number> STATEBUCKET=slalom0318-labuser<number> STATELOCKTABLE=slalom0318-labuser<number> STATEREGION=eu-west-2 make init

The expected output will contain the following

Created and switched to workspace "slalom0318-labuser12"!

## Terraform Plan and Apply

Before executing the code against the AWS API to create or modify resources you can check your code to make sure there are no errors. This is done by using the terraform plan function.

### Running the Plan - VPC

From the command line execute the following command, replace the <number> with your assigned labuser number. E.g labuser12.

STATE=vpc WORKSPACE=slalom0318-labuser<number> STATEBUCKET=slalom0318-labuser<number> STATELOCKTABLE=slalom0318-labuser<number> STATEREGION=eu-west-2 make plan

The excepted out will contain the following

**Plan:** 20 to add, 0 to change, 0 to destroy.

If there are no errors then continue you are good to apply the code.

### Running the Apply - VPC

The terraform apply will execute the code against the AWS API and begin to deploy the resources in your account.

From the command line execute the following command, replace the <number> with your assigned labuser number. E.g labuser12.

STATE=vpc WORKSPACE=slalom0318-labuser<number> STATEBUCKET=slalom0318-labuser<number> STATELOCKTABLE=slalom0318-labuser<number> STATEREGION=eu-west-2 make apply

Answer YES to the prompt ***Do you want to perform these actions?***

The excepted out will contain the following

Apply complete! Resources: 1 added, 1 changed, 0 destroyed.

Outputs: bastion\_dns = ec2-35-178-27-149.eu-west-2.compute.amazonaws.com

**NOTE : Make note of the bastion DNS name as you will need this later on in the lab**

### Running the Plan – APP

From the command line execute the following command, replace the <number> with your assigned labuser number. E.g labuser12.

STATE=app WORKSPACE=slalom0318-labuser<number> STATEBUCKET=slalom0318-labuser<number> STATELOCKTABLE=slalom0318-labuser<number> STATEREGION=eu-west-2 make plan

The excepted out will contain the following

Plan: 20 to add, 0 to change, 0 to destroy.

If there are no errors then continue you are good to apply the code.

### Running the Apply - APP

The terraform apply will execute the code against the AWS API and begin to deploy the resources in your account.

From the command line execute the following command, replace the <number> with your assigned labuser number. E.g labuser12.

STATE=app WORKSPACE=slalom0318-labuser<number> STATEBUCKET=slalom0318-labuser<number> STATELOCKTABLE=slalom0318-labuser<number> STATEREGION=eu-west-2 make apply

The excepted out will contain the following

Apply complete! Resources: 1 added, 1 changed, 0 destroyed.

**Outputs:**

app\_hostname = ip-172-18-16-4.eu-west-1.compute.internal

db\_hostname = ip-172-18-16-9.eu-west-1.compute.internal

web\_elb\_dns = labuser97-web-2035200460.eu-west-1.elb.amazonaws.com

**NOTE : Make note of the bastion DNS name as you will need this later on in the lab**

### Switch to the new bastion

You can now disconnect from the build server, type exit at the command prompt to drop the ssh session to the build server

Using SSH or putty connect to the bastion DNS name that you created earlier.

ssh -i "slalom-demo-lon.pem" ec2-user@<bastion\_dns>

**NOTE:** If you are using putty then follow the instructions from the start of the document but use the bastion DNS name in the **Host Name** field

e.g ec2-user@<bastion\_dns>

# running ansible

Ansible is a configuration management tool which runs over SSH. We will be using it to provision our AWS EC2 instances. Ansible is already preconfigured on the bastion instance through the EC2 UserData which is passed into the instance at point of creation.

Firstly, we will SSH onto the bastion server within our AWS environment, all configuration management will be run from this centralized location.

ssh -I /path/to/key/slalomdemo.pem ec2-user@bastion-hostname

We then need to create the private key on the bastion host to allow ansible to communicate with the other hosts in the stack. We will do this using the vim text editor running the following command below. From your local machine navigate to the lab materials folder

### \slalom-aws-lab\aws-demo-files\ssh keys\

Edit slalom-demo-lon.pem with a text editor, copy all the text to your clipboard – **NOTE**: these instructions leverage vim for all editing.

From the SSH terminal/putty enter the command below

vim ~/.ssh/demokey.pem

You will then need to press ‘I’ for Insert and paste in the key followed by pressing the ‘esc’ key and typing ‘:wq’ and press enter to save the file.

Give the SSH key the correct permissions:

chmod 400 ~/.ssh/demokey.pem

Clone the workshop repository to the home directory of the ec2-user

git clone https://github.com/nmarchini/tf-01.git

Navigate into the cloned folder and add the new hostnames of the app and db servers into our ansible inventory file (named hosts).

cd tf-01/ansible

vim hosts

You will need to update the IP address’s ‘127.0.0.1’ under the headers [appservers] and [dbservers] to the hostnames produced in the earlier terraform step e.g. ip-172-16-16-6.eu-west-2.compute.internal. You will then need to press ‘I’ for Insert and make the changes followed by ‘esc’ key and typing ‘:wq’ and enter to save the file.

The file should look like this once you are done (your hostnames will be different from the image below)



### 

Once our inventory is populated and our SSH key is in place, we are ready to run ansible for the first time. There are a number of options being passed in here with the ‘ansible-playbook’ command

* Playbook name: main.yml
* Private key: demokey.pem
* Username: ec2-user
* Inventory: hosts
* DB Hostname: the hostname of the mongodb server in your stack

ANSIBLE\_HOST\_KEY\_CHECKING=False ansible-playbook main.yml --private-key=~/.ssh/demokey.pem --user ec2-user -i hosts --extra-vars "db\_url=<DB\_HOSTNAME>"

Once the Ansible play book has finished the application and database will have been deployed. It will take a few minutes for the AWS load-balancer health check to become healthy so wait a little while before you move on to the final instruction.

**Test the site**

From a web browser on your local machine navigate to the DNS name of the web-elb that you noted down earlier. This will display the application that was just deployed.