



Terraform: Blue/Green Deployments with Lambda Functions

TENNIS SMITH

Table of Contents

Overview.....	2
Lab 1	2

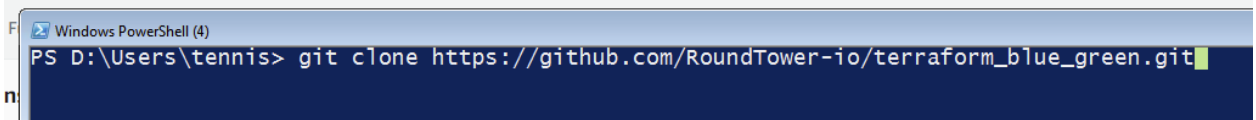
Overview

The purpose of this exercise is to demonstrate the use of blue/green deployments. We are using a lambda function to illustrate the point, but this could just as easily be done with individual ec2 instances.

We will create a website running the “blue” code. Then we will switch the website to “green” code.

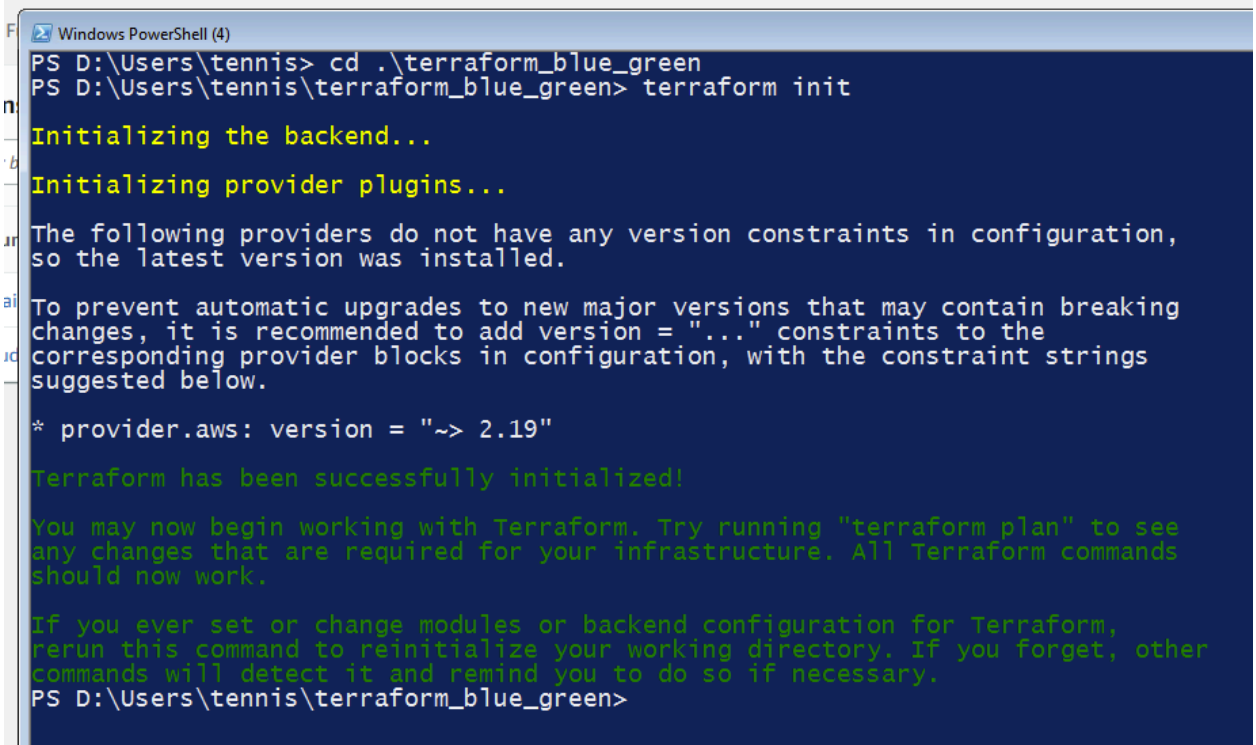
Lab 1

1. Open a shell in your WorkSpace
2. Clone a copy of https://github.com/RoundTower-io/terraform_blue_green.git



```
Windows PowerShell (4)
PS D:\Users\tennis> git clone https://github.com/RoundTower-io/terraform_blue_green.git
```

3. Change directory (“cd”) into the new directory and initialize Terraform



```
Windows PowerShell (4)
PS D:\Users\tennis> cd .\terraform_blue_green
PS D:\Users\tennis\terraform_blue_green> terraform init

Initializing the backend...

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.

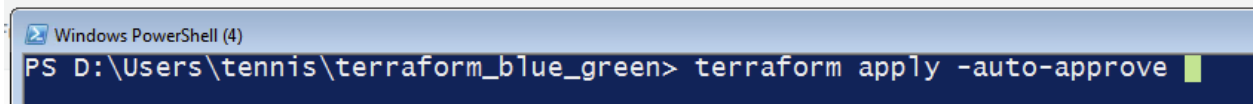
* provider.aws: version = "~> 2.19"

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.
PS D:\Users\tennis\terraform_blue_green>
```

4. Do “terraform apply -auto-approve” to create the “blue” version



```
Windows PowerShell (4)
PS D:\Users\tennis\terraform_blue_green> terraform apply -auto-approve
```

5. Notice the final lines in the output will look something like this

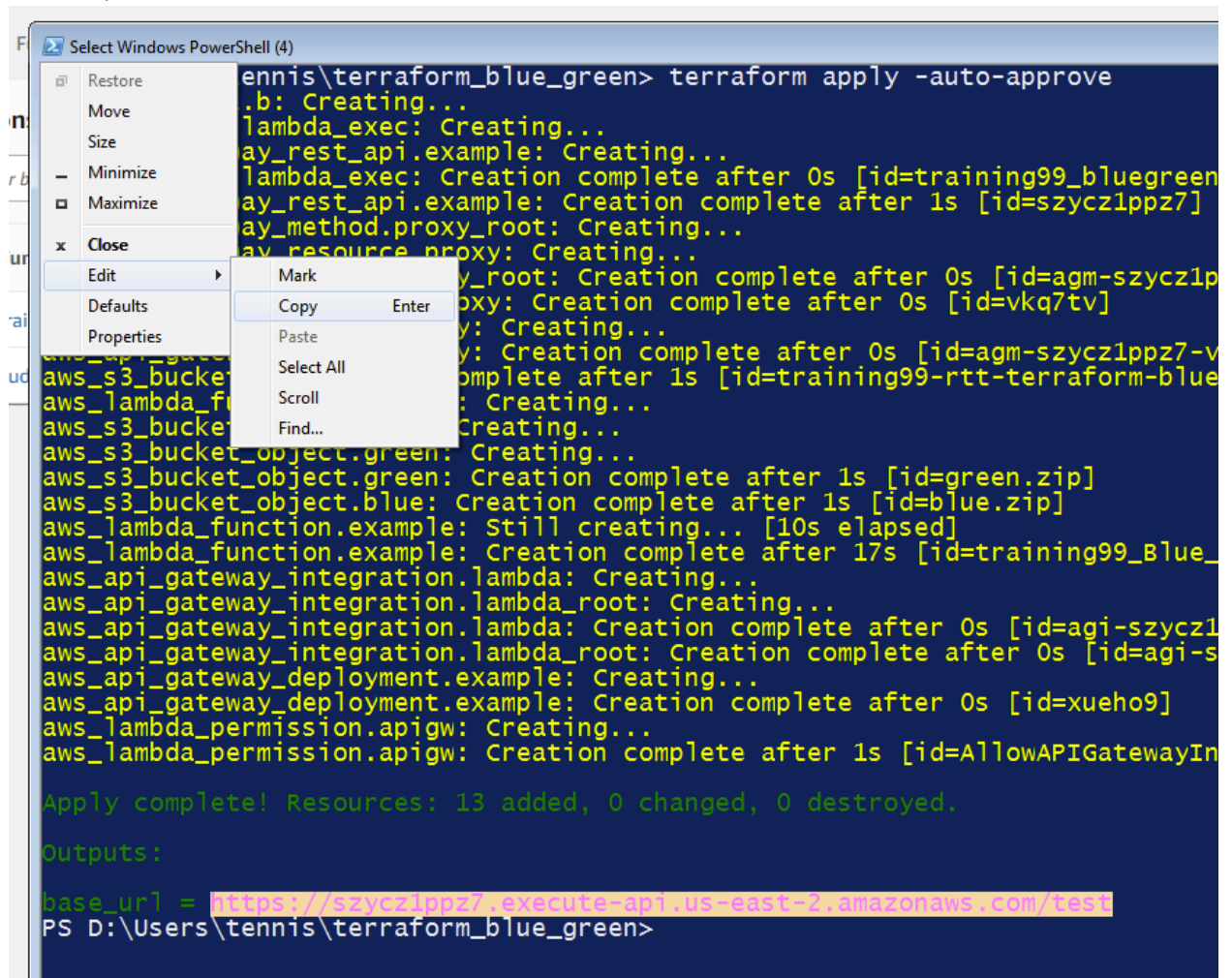
```
aws_lambda_permission.apigw: Creating...
aws_lambda_permission.apigw: Creation complete after 1s [id=AllowAPIGatewayInvoke]

Apply complete! Resources: 13 added, 0 changed, 0 destroyed.

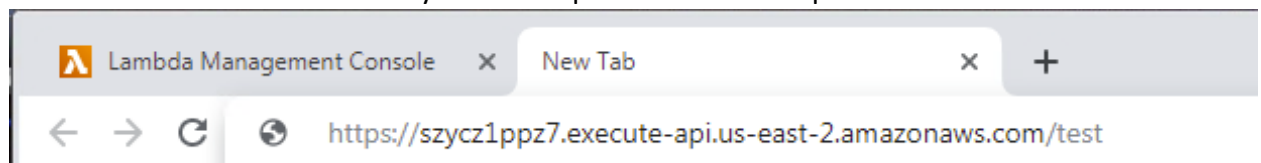
Outputs:

base_url = https://szycz1ppz7.execute-api.us-east-2.amazonaws.com/test
PS D:\Users\tennis\terraform_blue_green>
```

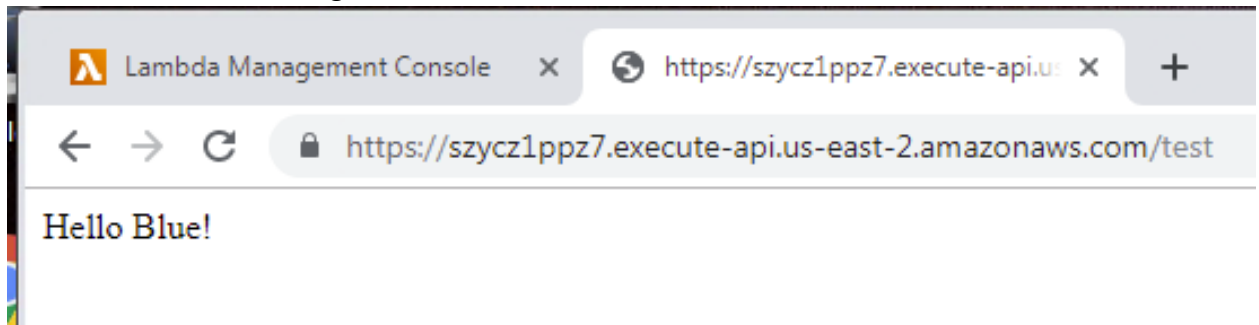
6. The URL listed will be the URL of your website.
7. Copy the URL. To do that, highlight the URL in the terminal with your mouse. Then press <enter>, OR use the menu as shown below.



8. Paste the URL into a new tab in your Workspace browser and press <enter>



9. You should see something like this



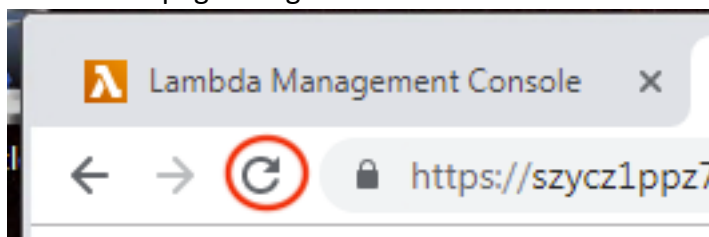
10. Now we want to implement our “green” code version. To do that, go back to your terminal and type ‘`terraform apply -var="deployment=green" -auto-approve`’ and press <enter>

```
PS D:\Users\tennis\terraform_blue_green> terraform apply -var="deployment=green" -auto-approve
aws_iam_role.lambda_exec: Refreshing state... [id=training99_bluegreen_example_lambda]
aws_api_gateway_rest_api.example: Refreshing state... [id=szycz1ppz7]
aws_s3_bucket.b: Refreshing state... [id=training99-rtt-terraform-blue-green-example]
aws_api_gateway_resource.proxy: Refreshing state... [id=vkq7tv]
aws_api_gateway_method.proxy_root: Refreshing state... [id=agm-szycz1ppz7-jn0f2udv3m-ANY]
aws_api_gateway_method.proxy: Refreshing state... [id=agm-szycz1ppz7-vkq7tv-ANY]
aws_s3_bucket_object.blue: Refreshing state... [id=blue.zip]
aws_s3_bucket_object.green: Refreshing state... [id=green.zip]
aws_lambda_function.example: Refreshing state... [id=training99_Blue_Green]
aws_api_gateway_integration.lambda: Refreshing state... [id=agi-szycz1ppz7-vkq7tv-ANY]
aws_api_gateway_integration.lambda_root: Refreshing state... [id=agi-szycz1ppz7-jn0f2udv3m-ANY]
aws_api_gateway_deployment.example: Refreshing state... [id=xueho9]
aws_lambda_permission.apigw: Refreshing state... [id=AllowAPIGatewayInvoke]
aws_lambda_function.example: Modifying... [id=training99_Blue_Green]
aws_lambda_function.example: Modifications complete after 1s [id=training99_Blue_Green]

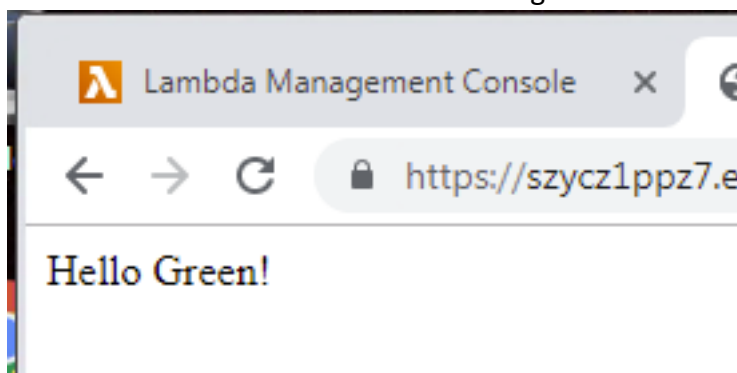
Apply complete! Resources: 0 added, 1 changed, 0 destroyed.

Outputs:
base_url = https://szycz1ppz7.execute-api.us-east-2.amazonaws.com/test
PS D:\Users\tennis\terraform_blue_green>
```

11. Go back to your browser to your URL tab
12. Refresh the page using the refresh icon



13. Your browser should now look something like this



14. Tear down your website by doing “`terraform destroy -auto-approve`”

15. Make sure it completes. The final like of the output will look something like this

```
aws_iam_role.lambda_exec: Destruction complete  
aws_s3_bucket.b: Destruction complete after 1s  
aws_api_gateway_rest_api.example: Destruction  
  
Destroy complete! Resources: 13 destroyed.  
PS D:\Users\tennis\terraform_blue_green>
```