
PROJECT DETAILS

Project Title

Deployment of WordPress Environment using CMAT

Problem statement

You are a DevOps engineer at SL Ltd. Your company is working mostly on WordPress projects. A lot of development hours are lost to perform WordPress setup with all dependencies like PHP, MySQL, etc. The Company wants to automate it with the help of a configuration management tool so that they can follow a standard installation procedure for WordPress and its components whenever a new requirement or client comes in. The below mentioned components should be included:

- PHP
- Nginx/Apache Web Server
- MySQL
- WordPress

Tool required

- Terraform (installed on a controller machine)
- Ansible controller (installed on a controller machine)
- AWS account with security credentials
- For SSH I have created in terraform command code put ssh-key with instance that will create in future
- PHP
- MYSQL
- WORDPRESS
- APACHE2/NGINX

Need configuration on instance after created

- Launch an EC2 instance using Terraform (linkup ssh-key pair)
- Terraform host will connect created ec2 instance via ssh-key-pair.
- Run playbook and install required tools on ec2 instance.
- Setup WordPress on created ec2 instance.
- Install php, MySQL, apache2/nginx and Python in the instance.
- Validate that the packages are installed with WordPress running.

Procedure

Step 1. Configure the controller node

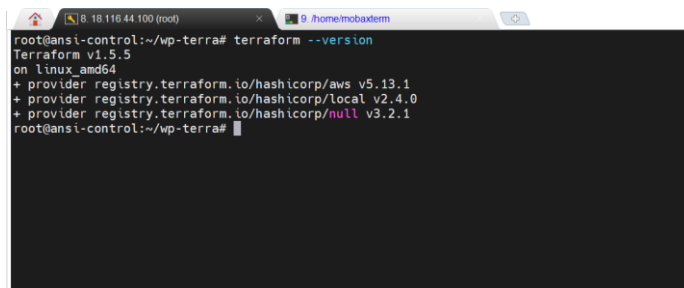
Installation steps:

➤ Procedure to install terraform in controller:

- First need to visit
https://developer.hashicorp.com/terraform/downloads?product_intent=terraform
- Follow the below command to install terraform in linux os at controller node

```
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg
--dearmor -o /usr/share/keyrings/hashicorp-archive-
keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-
keyring.gpg] https://apt.releases.hashicorp.com
$(lsb_release -cs) main" | sudo tee
/etc/apt/sources.list.d/hashicorp.list
sudo apt update && sudo apt install terraform
```

- To validate:
 - Terraform version
 - Which terraform

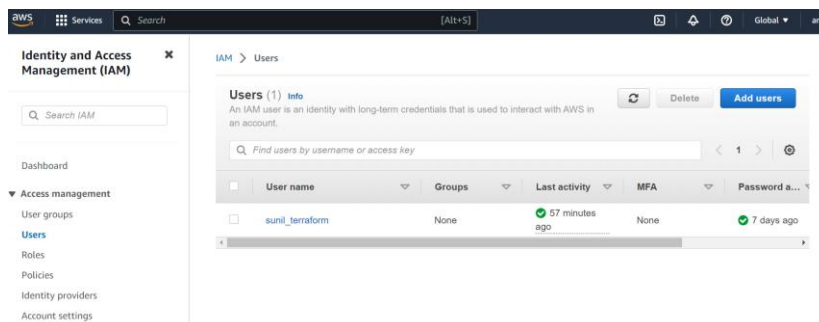


```
root@ansi-control:~/wp-terra# terraform --version
Terraform v1.5.5
on linux_amd64
+ provider registry.terraform.io/hashicorp/aws v5.13.1
+ provider registry.terraform.io/hashicorp/local v2.4.0
+ provider registry.terraform.io/hashicorp/null v3.2.1
root@ansi-control:~/wp-terra#
```

Step 2: Setup AWS “access key” & “secret key” (same followed with below setup)

➤ Procedure to setup access key and secret key:

- Visit AWS account > go to IAM service > user section > create user



- Click “Add users” in case it not created in my case it is already created

Specify user details

User details

User name
sunil1

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and +, -, @, _ (hyphen)

☒ **Provide user access to the AWS Management Console - optional**
If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

Are you providing console access to a person?

User type

☒ **Specify a user in Identity Center - Recommended**
We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.

☐ I want to create an IAM user

- Click “I want to create an IAM user” and then then follow as below

Console password

☒ **Autogenerated password**
You can view the password after you create the user.

☐ **Custom password**
Enter a custom password for the user.

Must be at least 8 characters long
Must include at least three of the following mix of character types: uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), and symbols ! @ # \$ % ^ & * () _ + - (hyphen) = [] { } |

☐ Show password

☐ **Users must create a new password at next sign-in - Recommended**
Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.

Information If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

Cancel Next

- Click “next” & attach policy as “AdministratorAccess”

Permissions options

☐ **Add user to group**
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ **Copy permissions**
Copy all group memberships, attached managed policies, and inline policies from an existing user.

☒ **Attach policies directly**
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1106) [Refresh](#) [Create policy](#)

Choose one or more policies to attach to your new user.

Filter by Type
admin X All types 37 matches < 1 2 > ⓘ

	Policy name	Type	Attached entities
<input checked="" type="checkbox"/>	AdministratorAccess	AWS managed - job function	1
<input type="checkbox"/>	AdministratorAccess-A...	AWS managed	0
<input type="checkbox"/>	AdministratorAccess-A...	AWS managed	0

- Now create user

sunil1 Autogenerated No

Permissions summary

< 1 >

Name	Type	Used as
AdministratorAccess	AWS managed - job function	Permissions policy

Tags - optional

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

Cancel Previous **Create user**

Retrieve password

You can view and download the user's password below or email users instructions for signing in to the AWS Management Console. This is the only time you can view and download this password.

Console sign-in details

[Email sign-in instructions](#)

Console sign-in URL
<https://763351537969.signin.aws.amazon.com/console>

User name
 sunil1

Console password
 ***** [Show](#)

[Download .csv file](#) **Return to users list**

- Now create access and secret key for above user

Summary

ARN arn:aws:iam::763351537969:user:sunil1	Console access Enabled without MFA	Access key 1 Not enabled
Created July 01, 2023, 19:00 (UTC+05:30)	Last console sign-in Never	Access key 2 Not enabled

Permissions Groups Tags **Security credentials** Access Advisor

Console sign-in

[Manage console access](#)

Console sign-in link
<https://763351537969.signin.aws.amazon.com/console>

Console password
 Updated 1 minute ago (2023-07-01 19:00 GMT+5:30)

- Click create access key

Remove Resync **Assign MFA device**

Device type	Identifier	Certifications	Created on
No MFA devices. Assign an MFA device to improve the security of your AWS environment			

[Assign MFA device](#)

Access keys (0)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

Create access key

No access keys

As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. [Learn more](#)

[Create access key](#)

- Select for command line

Access key best practices & alternatives [Info](#)

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

☒ **Command Line Interface (CLI)**
 You plan to use this access key to enable the AWS CLI to access your AWS account.

☐ **Local code**
 You plan to use this access key to enable application code in a local development environment to access your AWS account.

☐ **Application running on an AWS compute service**
 You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

☐ **Third-party service**
 You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

- Check policy and click next

☐ **Third-party service**
 You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

☐ **Application running outside AWS**
 You plan to use this access key to enable an application running on an on-premises host, or to use a local AWS client or third-party AWS plugin.

☐ **Other**
 Your use case is not listed here.

Alternatives recommended

- Use [AWS CloudShell](#), a browser-based CLI, to run commands. [Learn more](#)
- Use the [AWS CLI V2](#) and enable authentication through a user in IAM Identity Center. [Learn more](#)

Confirmation

☒ I understand the above recommendation and want to proceed to create an access key.

Cancel Next

- Access and secret key generated and need to save for future used

Retrieve access keys [Info](#)

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIAIOSFODNN7EXAMPLE	***** Show

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.

Step3 Setup wordpress via terraform and ansible.

- Procedure to follow as mentioned below:
 - First need to create directory as “wp-terra” in my case
 - Use command “mkdir wp-terra” && “cd wp-terra”

```
root@ansi-control:~/wp-terra#  
root@ansi-control:~/wp-terra#  
root@ansi-control:~/wp-terra#  
root@ansi-control:~/wp-terra#
```

- I have used roles for install dependent tools like (php, apache, mysql, wordpress)
- So in project directory I have created roles, inside roles I have installed roles as below

```
root@ansi-control:~/wp-terra/roles# ll  
total 24  
drwxr-xr-x 6 root root 4096 Aug 20 06:00 ./  
drwxr-xr-x 6 root root 4096 Aug 20 08:25 ../  
drwxr-xr-x 10 root root 4096 Aug 20 05:38 mysql/  
drwxr-xr-x 10 root root 4096 Aug 20 05:38 php/  
drwxr-xr-x 10 root root 4096 Aug 20 05:38 server/  
drwxr-xr-x 10 root root 4096 Aug 20 05:38 wordpress/  
root@ansi-control:~/wp-terra/roles#
```

- On project directory, I have to create terraform files and ansible-playbook to execute different tools by roles on ec2 instance.

```
root@ansi-control:~/wp-terra# ll  
total 64  
drwxr-xr-x 6 root root 4096 Aug 20 08:25 ./  
drwx----- 14 root root 4096 Aug 20 09:38 ../  
drwxr-xr-x 8 root root 4096 Aug 20 08:35 .git/  
drwxr-xr-x 3 root root 4096 Aug 20 06:44 .terraform/  
-rw-r--r-- 1 root root 3468 Aug 20 06:44 .terraform.lock.hcl  
-rw-r--r-- 1 root root 1668 Aug 20 08:00 main.tf  
-rw-r--r-- 1 root root 102 Aug 20 08:09 playbook.yml  
-rw-r--r-- 1 root root 174 Aug 20 08:01 provider.tf  
drwxr-xr-x 6 root root 4096 Aug 20 06:00 roles/  
-rw-r--r-- 1 root root 593 Aug 20 06:46 sg.tf  
drwxr-xr-x 2 root root 4096 Aug 20 06:37 templates/  
-rw-r--r-- 1 root root 181 Aug 20 08:21 terraform.tfstate  
-rw-r--r-- 1 root root 10886 Aug 20 08:20 terraform.tfstate.backup  
-rw-r--r-- 1 root root 232 Aug 20 08:01 variable.tf  
root@ansi-control:~/wp-terra#
```

- Vi provider.tf >> press "i" insert for write then write code as mention below

```
• provider "aws" {  
•   profile = "skadm"  
•   region  = "us-east-1"  
• }  
•  
• resource "aws_key_pair" "key-tfnew" {  
•   key_name     = "key-tfnew"  
•   public_key   = file("~/ssh/id_rsa.pub")  
• }  
•
```

- Then press “:wq!” to save

- Variable.tf files look like below

```
variable "aws_ami_id" {
  # ami for ubuntu
  default = "ami-0261755bbcb8c4a84"
  ## "ami-0261755bbcb8c4a84"
}
variable "ssh_key_pair" {
  default = "~/.ssh/id_rsa"
}
variable "ssh_key_pair_pub" {
  default = "~/.ssh/id_rsa.pub"
}
```

- Security_group.tf file shown below:

```
resource "aws_security_group" "skv-terra" {
  name           = "skv-tf3"
  description    = "Allow http https and ssh traffic via terraform"
  ingress {
    from_port     = 80
    to_port       = 80
    protocol      = "tcp"
    cidr_blocks   = ["0.0.0.0/0"]
  }
  ingress {
    from_port     = 443
    to_port       = 443
    protocol      = "tcp"
    cidr_blocks   = ["0.0.0.0/0"]
  }
  ingress {
    from_port     = 22
    to_port       = 22
    protocol      = "tcp"
    cidr_blocks   = ["0.0.0.0/0"]
  }
  egress {
    from_port     = 0
    to_port       = 0
    protocol      = "-1"
    cidr_blocks   = ["0.0.0.0/0"]
  }
}
```

- **Hosts file**

```

≡ hosts
1  [wordpress]
2  ${public_ipaddr}
3
4  [wordpress:vars]
5  ansible_ssh_user = ubuntu
6  ansible_ssh_private_key_file = ${key_path}

```

- **Main.tf file look like below:**

```

• # Main Terraform configuration file
•
• # Define the local variables for the root module
• locals {
•   ami_id = "ami-0261755bbcb8c4a84"
•   ssh_user = "ubuntu"
•   key_name = "wpserver"
•   private_key_path = "${path.module}/wpserver.pem"
• }
•
• # This creates the EC2 instance and makes an initial SSH
• connection.
• resource "aws_instance" "wpserver" {
•   ami = var.aws_ami_id
•   instance_type = "t2.micro"
•   associate_public_ip_address = true
•   vpc_security_group_ids = [aws_security_group.skv-terra.id]
•   key_name = "key-tfnew"
•
•   tags = {
•     Name = "WordPress Server"
•   }
•
•   connection {
•     type = "ssh"
•     host = self.public_ip
•     user = local.ssh_user
•     private_key = file(var.ssh_key_pair)
•     timeout = "4m"
•   }
•
•   provisioner "remote-exec" {
•     inline = [
•       "touch /home/ubuntu/demo-file-from-terraform.txt"
•     ]
•   }
• }

```



```

•   }
•   }
•
•   # Creating a local hosts file for Ansible to use
•   resource "local_file" "hosts" {
•       content = templatefile("${path.module}/templates/hosts",
•       {
•           public_ipaddr = aws_instance.wpserver.public_ip
•           key_path = var.ssh_key_pair
•           ansible_user = local.ssh_user
•       }
•       )
•       filename = "${path.module}/hosts"
•   }
•
•   # We will use a null resource to execute the playbook with a
•   local-exec provisioner.
•
•   resource "null_resource" "run_playbook" {
•       depends_on = [
•
•           # Running of the playbook depends on the successful
•           creation of the EC2
•           # instance and the local inventory file.
•
•           aws_instance.wpserver,
•           local_file.hosts,
•       ]
•
•       provisioner "local-exec" {
•           command = "ANSIBLE_HOST_KEY_CHECKING=False ansible-
•           playbook -u ubuntu -i hosts --private-key ${var.ssh_key_pair}
•           -e 'pub_key=${var.ssh_key_pair_pub}' playbook.yml"
•       }
•   }
•
•
•

```

- Now I have setup all prerequisites and installed terraform and ansible in controller
To run ansible-playbook install wordpress.
- Now I have initiated terraform "terraform init".


```
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exact
terraform apply now.
root@ansi-control:~/wp-terra#
root@ansi-control:~/wp-terra# terraform apply
Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_instance.wpserver will be created
+ resource "aws_instance" "wpserver" {
  + ami                    = "ami-0261755bbc8c4a84"
  + arn                   = (known after apply)
  + associate_public_ip_address = true
  + availability_zone      = (known after apply)
  + cpu_core_count         = (known after apply)
  + cpu_threads_per_core   = (known after apply)
  + disable_api_stop       = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized          = false
  + get_password_data      = (known after apply)
  + host_id                = (known after apply)
  + host_resource_group_arn = (known after apply)
  + iam_instance_profile   = (known after apply)
  + id                     = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)

# local_file.hosts will be created
+ resource "local_file" "hosts" {
  + content              = (known after apply)
  + content_base64sha256 = (known after apply)
  + content_base64sha512 = (known after apply)
  + content_md5          = (known after apply)
  + content_sha1         = (known after apply)
  + content_sha256       = (known after apply)
  + content_sha512       = (known after apply)
  + directory_permission = "0777"
  + file_permission      = "0777"
  + filename             = "./hosts"
  + id                   = (known after apply)

# null_resource.run_playbook will be created
+ resource "null_resource" "run_playbook" {
  + id = (known after apply)
}

Plan: 5 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: 
```

- Put “yes” to proceed creation instance and run playbook to install wordpress setup.

```
+ resource "local_file" "hosts" {
  + content              = (known after apply)
  + content_base64sha256 = (known after apply)
  + content_base64sha512 = (known after apply)
  + content_md5          = (known after apply)
  + content_sha1         = (known after apply)
  + content_sha256       = (known after apply)
  + content_sha512       = (known after apply)
  + directory_permission = "0777"
  + file_permission      = "0777"
  + filename             = "./hosts"
  + id                   = (known after apply)
}

# null_resource.run_playbook will be created
+ resource "null_resource" "run_playbook" {
  + id = (known after apply)
}

Plan: 5 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_security_group.skv-terra: Creating...
aws_key_pair.key-tfnew: Creating...
aws_key_pair.key-tfnew: Creation complete after 0s [id=key-tfnew]
aws_security_group.skv-terra: Creation complete after 2s [id=sg-650d3345cdc295d20]
aws_instance.wpserver: Creating...
```

- support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

```
aws_instance.wpserver (remote-exec): User: ubuntu
aws_instance.wpserver (remote-exec): Password: false
aws_instance.wpserver (remote-exec): Private key: true
aws_instance.wpserver (remote-exec): Certificate: false
aws_instance.wpserver (remote-exec): SSH Agent: false
aws_instance.wpserver (remote-exec): Checking Host Key: false
aws_instance.wpserver (remote-exec): Target Platform: unix
aws_instance.wpserver (remote-exec): Connecting to remote host via SSH...
aws_instance.wpserver (remote-exec): Host: 54.209.36.91
aws_instance.wpserver (remote-exec): User: ubuntu
aws_instance.wpserver (remote-exec): Password: false
aws_instance.wpserver (remote-exec): Private key: true
aws_instance.wpserver (remote-exec): Certificate: false
aws_instance.wpserver (remote-exec): SSH Agent: false
aws_instance.wpserver (remote-exec): Checking Host Key: false
aws_instance.wpserver (remote-exec): Target Platform: unix
aws_instance.wpserver (remote-exec): Connected!
aws_instance.wpserver: Still creating... [100s elapsed]
aws_instance.wpserver: Creation complete after 1m1s [id=i-03788c15fc6bdf74a]
local_file.hosts: Creating...
local_file.hosts: Creation complete after 0s [id=61ca35de689fe159a709247a07f6ab2c875522a]
null_resource.run_playbook: Creating...
null_resource.run_playbook: Provisioning with 'local-exec'...
null_resource.run_playbook (local-exec): Executing: ["bash" "-c" "ANSIBLE_HOST_KEY_CHECKING=False ansible-playbook -u ubuntu -i hosts --private-key ~/.ssh/id_rsa -e 'pub_key=~/.ssh/id_rsa.pub' playbook.yml"]
null_resource.run_playbook (local-exec): PLAY [wordpress] *****
null_resource.run_playbook (local-exec): TASK [server : Update apt cache] *****
null_resource.run_playbook (local-exec): Still creating... [10s elapsed]
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
```

```
local_file.hosts: Creating...
local_file.hosts: Creation complete after 0s [id=61ca35de689fe159a709247a07f6ab2c875522a]
null_resource.run_playbook: Creating...
null_resource.run_playbook: Provisioning with 'local-exec'...
null_resource.run_playbook (local-exec): Executing: ["bash" "-c" "ANSIBLE_HOST_KEY_CHECKING=False ansible-playbook -u ubuntu -i hosts --private-key ~/.ssh/id_rsa -e 'pub_key=~/.ssh/id_rsa.pub' playbook.yml"]
null_resource.run_playbook (local-exec): PLAY [wordpress] *****
null_resource.run_playbook (local-exec): TASK [server : Update apt cache] *****
null_resource.run_playbook (local-exec): Still creating... [10s elapsed]
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
null_resource.run_playbook (local-exec): TASK [server : Install required software] *****
null_resource.run_playbook (local-exec): Still creating... [20s elapsed]
null_resource.run_playbook (local-exec): Still creating... [30s elapsed]
null_resource.run_playbook (local-exec): Still creating... [40s elapsed]
null_resource.run_playbook (local-exec): Still creating... [50s elapsed]
null_resource.run_playbook (local-exec): Still creating... [1m0s elapsed]
null_resource.run_playbook (local-exec): Still creating... [1m20s elapsed]
null_resource.run_playbook (local-exec): Still creating... [1m40s elapsed]
null_resource.run_playbook (local-exec): Still creating... [1m40s elapsed]
```

```
null_resource.run_playbook: Still creating... [1m40s elapsed]
null_resource.run_playbook: Still creating... [1m50s elapsed]
null_resource.run_playbook: Still creating... [2m0s elapsed]
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
null_resource.run_playbook (local-exec): TASK [php : Install php extensions] *****
null_resource.run_playbook: Still creating... [2m10s elapsed]
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
null_resource.run_playbook (local-exec): TASK [mysql : Create mysql database] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
null_resource.run_playbook (local-exec): TASK [mysql : Create mysql user] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
null_resource.run_playbook (local-exec): TASK [wordpress : Download WordPress] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
null_resource.run_playbook (local-exec): TASK [wordpress : Extract WordPress] *****
null_resource.run_playbook: Still creating... [2m20s elapsed]
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
null_resource.run_playbook (local-exec): TASK [wordpress : Update default Apache site] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
null_resource.run_playbook (local-exec): TASK [wordpress : Copy sample config file] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
null_resource.run_playbook (local-exec): TASK [wordpress : Wordpress | Copy wp-config.php file content] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
```

```
null_resource.run_playbook (local-exec): TASK [mysql : Create mysql database] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
null_resource.run_playbook (local-exec): TASK [mysql : Create mysql user] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
null_resource.run_playbook (local-exec): TASK [wordpress : Download WordPress] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
null_resource.run_playbook (local-exec): TASK [wordpress : Extract WordPress] *****
null_resource.run_playbook: Still creating... [2m20s elapsed]
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
null_resource.run_playbook (local-exec): TASK [wordpress : Update default Apache site] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
null_resource.run_playbook (local-exec): TASK [wordpress : Copy sample config file] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
null_resource.run_playbook (local-exec): TASK [wordpress : Wordpress | Copy wp-config.php file content] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]
null_resource.run_playbook (local-exec): RUNNING HANDLER [wordpress : restart apache] *****
null_resource.run_playbook (local-exec): PLAY RECAP *****
null_resource.run_playbook (local-exec): 54.209.36.91 : ok=11 changed=11 unreachable=0 failed=0 skip=
d=0 rescued=0 ignored=0
null_resource.run_playbook: Creation complete after 2m26s [id=7355569171488225460]
Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
root@msi-control:~/wp-terra#
```

- Now terraform has created ec2 instance with below public ip
 - Public ip

```

null_resource.run_playbook (local-exec): TASK [mysql: Create mysql database] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]

null_resource.run_playbook (local-exec): TASK [mysql: Create mysql user] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]

null_resource.run_playbook (local-exec): TASK [wordpress: Download WordPress] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]

null_resource.run_playbook (local-exec): TASK [wordpress: Extract WordPress] *****
null_resource.run_playbook (local-exec): Still creating... [2m20s elapsed]
null_resource.run_playbook (local-exec): changed: [54.209.36.91]

null_resource.run_playbook (local-exec): TASK [wordpress: Update default Apache site] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]

null_resource.run_playbook (local-exec): TASK [wordpress: Copy sample config file] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]

null_resource.run_playbook (local-exec): TASK [wordpress: WordPress | Copy wp-config.php file content] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]

null_resource.run_playbook (local-exec): RUNNING HANDLER [wordpress: restart apache] *****
null_resource.run_playbook (local-exec): changed: [54.209.36.91]

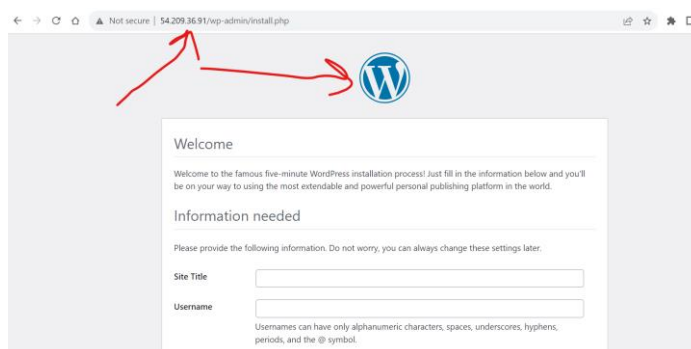
null_resource.run_playbook (local-exec): PLAY RECAP *****
null_resource.run_playbook (local-exec): 54.209.36.91 : ok=11 changed=11 unreachable=0 failed=0 skipped=0
rescued=0 ignored=0

null_resource.run_playbook (local-exec): Creation complete after 2m26s [id=7355569171488225460]

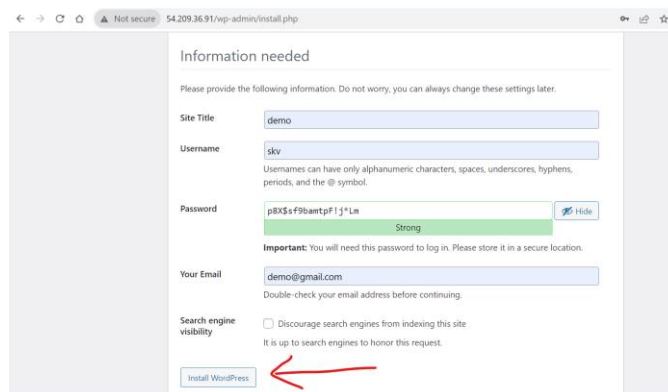
Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
root@ms-control:~# wp-terraf

```

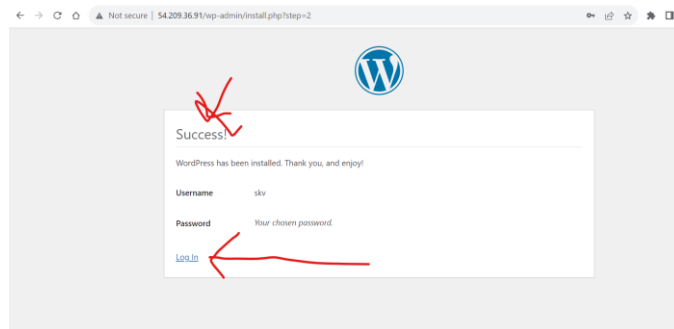
- Now take this public ip and put in web browser and see the magic
- Wow!!!!... wordpress is installed on newly created instance as below:



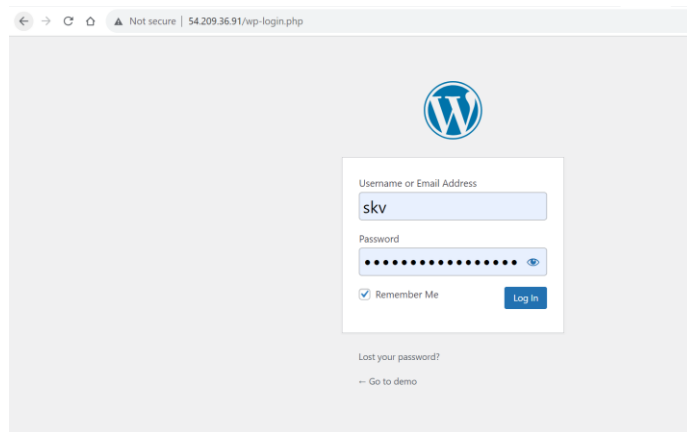
- Now setup the wordpress with enter few required details. And then click install wordpress.



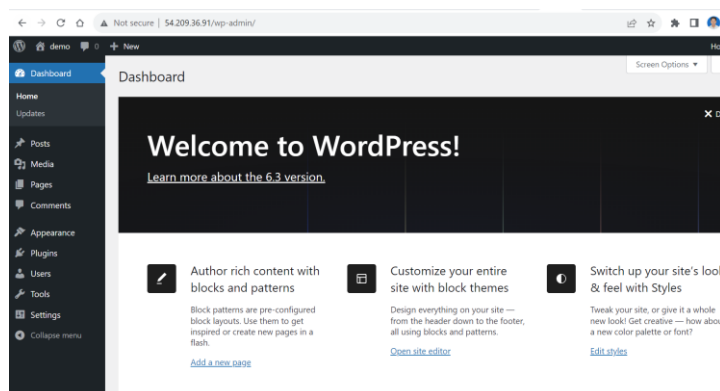
- To check if data base is accessible the put above details and look below snip.



- Same Followed below:



- Now I have setup wordpress and it looks like below:



Documents for project at github:

- Documents for wordpress:
My github project link "<https://github.com/vermasunilk792/cmat-proj2-sl.git>"