

## Part – 1: Initial Configuration

*Fig 1.1: Changing execution\_role\_arn in ecs.tf file*

```

# ECS Cluster
resource "aws_ecs_cluster" "cluster" {
  name = var.ecs_cluster_name
}

# ECS Task Definition
resource "aws_ecs_task_definition" "task" {
  family      = "wp-task"
  container_definitions = data.template_file.wp.container.rendered
  network_mode = "awsvpc"
  requires_compatibilities = ["FARGATE"]
  cpu          = "2048"
  memory       = "8192"
  execution_role_arn = "arn:aws:iam::357856359823:role/LabRole" # Use the existing role ARN
}

# ECS Service
resource "aws_ecs_service" "service" {
  name        = "wp-service"
  cluster     = aws_ecs_cluster.cluster.id
  task_definition = aws_ecs_task_definition.task.arn
  desired_count = "1"
  launch_type = "FARGATE"

  network_configuration {
    subnets = [aws_subnet_wp_public_a_tf.id, aws_subnet_wp_public_b_tf.id, aws_subnet_wp_public_c_tf.id]
    security_groups = [aws_security_group_wp_alb_tf.id]
    assign_public_ip = true
  }

  load_balancer {
    target_group_arn = aws_lb_target_group_default_tf.arn
    container_name   = "wordpress"
    container_port   = 80
  }
}

depends_on = [aws_lb_listener.default]
}

```

**Note:** From the Terraform setup files, the ecs.tf file by default has the execution\_role\_arn value set to a previous user. Update this with your own execution\_role\_arn, which you can find in the AWS Dashboard under your IAM role.

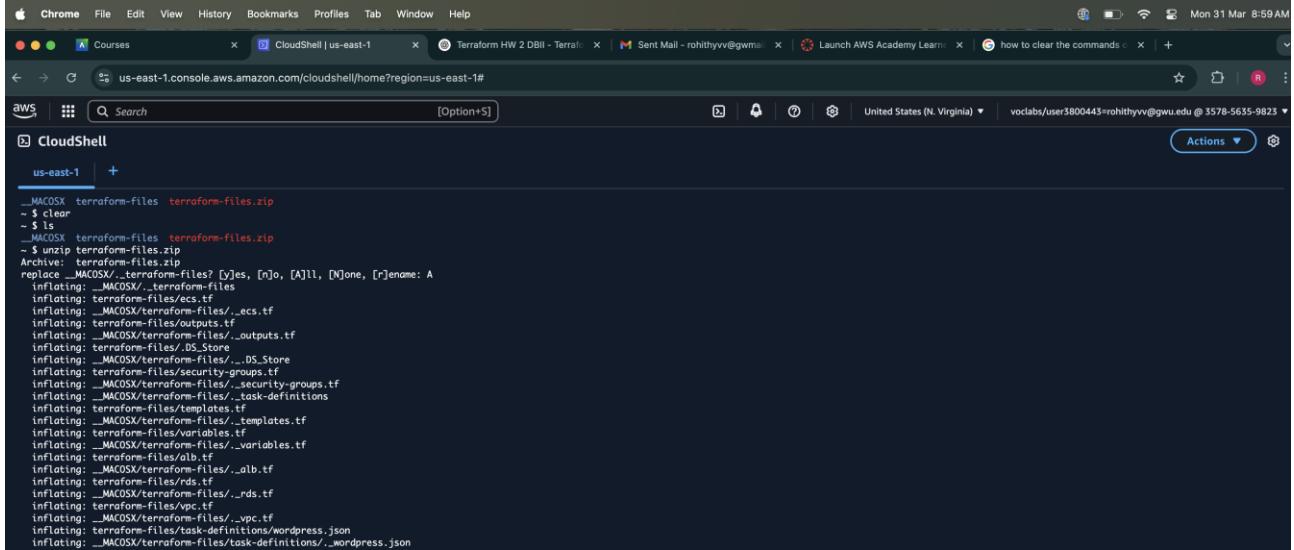
### Key Note:

From the terraform setup files, update the execution\_role\_arn in ecs.tf file with our own IAM user ARN from AWS dashboard to allow ECS to access AWS services securely.

This step ensures that the ECS task gets the right execution permissions and doesn't fail due to using a previous user's IAM role.

## Part – 2: Infrastructure Deployment Using Terraform

*Fig 2.1: Uploading the terraform zip files to the AWS*

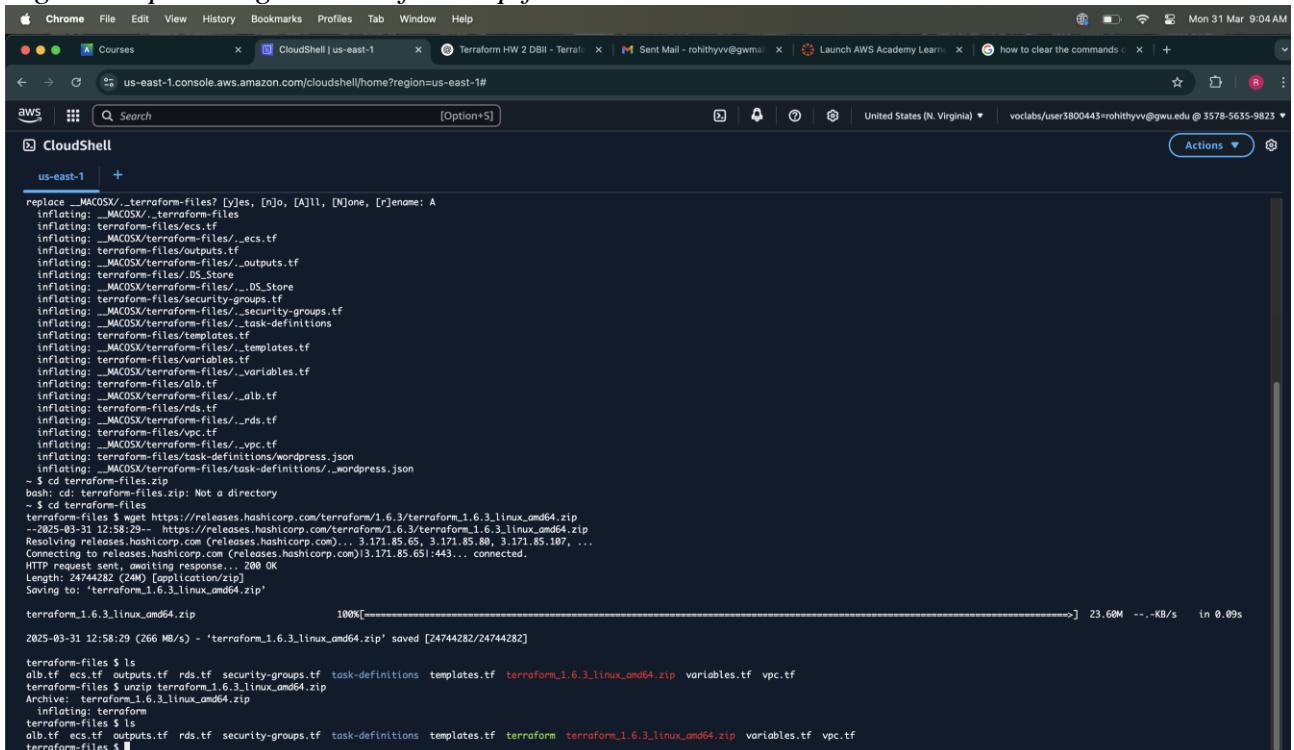


The screenshot shows a Chrome browser window with multiple tabs open. The active tab is 'CloudShell | us-east-1'. The terminal window displays the command `aws s3 cp terraform-files.zip s3://...` followed by a list of files being uploaded from a 'terraform-files.zip' archive. The files listed include various Terraform configuration files like `ecs.tf`, `outputs.tf`, `security-groups.tf`, and `vpc.tf`, along with `variables.tf` and `task-definitions/wordpress.json`.

```
_MACOSX_terraform-files.zip
$ clear
$ ls
$ aws s3 cp terraform-files.zip s3://...
Archive: terraform-files.zip
replace _MACOSX_terraform-files? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
inflating: _MACOSX_terraform-files
inflating: terraform-files/ecs.tf
inflating: terraform-files/outputs.tf
inflating: terraform-files/_DS_Store
inflating: _MACOSX_terraform-files/_DS_Store
inflating: terraform-files/security-groups.tf
inflating: _MACOSX_terraform-files/_security-groups.tf
inflating: _MACOSX_terraform-files/_task-definitions
inflating: terraform-files/templates.tf
inflating: _MACOSX_terraform-files/_templates.tf
inflating: terraform-files/variables.tf
inflating: _MACOSX_terraform-files/_variables.tf
inflating: terraform-files/alb.tf
inflating: _MACOSX_terraform-files/_alb.tf
inflating: terraform-files/rds.tf
inflating: _MACOSX_terraform-files/_rds.tf
inflating: terraform-files/vpc.tf
inflating: _MACOSX_terraform-files/_vpc.tf
inflating: terraform-files/task-definitions/wordpress.json
inflating: _MACOSX_terraform-files/task-definitions/_wordpress.json
```

**Note:** The Terraform zip file is uploaded into AWS CloudShell and is ready to be unzipped for the next steps.

*Fig 2.2: Uploading the terraform zip files to the AWS*



The screenshot shows a Chrome browser window with multiple tabs open. The active tab is 'CloudShell | us-east-1'. The terminal window displays the command `aws s3 cp terraform-files.zip s3://...` followed by a list of files being uploaded from a 'terraform-files.zip' archive. The files listed are identical to Fig 2.1. Below this, the command `cd terraform-files` is run, followed by `ls` which shows the contents of the directory including `alb.tf`, `ecs.tf`, `outputs.tf`, `rds.tf`, `security-groups.tf`, `task-definitions`, `templates.tf`, `variables.tf`, and `vpc.tf`. The terminal then shows the download of 'terraform\_1.6.3\_linux\_amd64.zip' from HashiCorp's website, a successful extraction of the file, and a final `ls` command showing the extracted files.

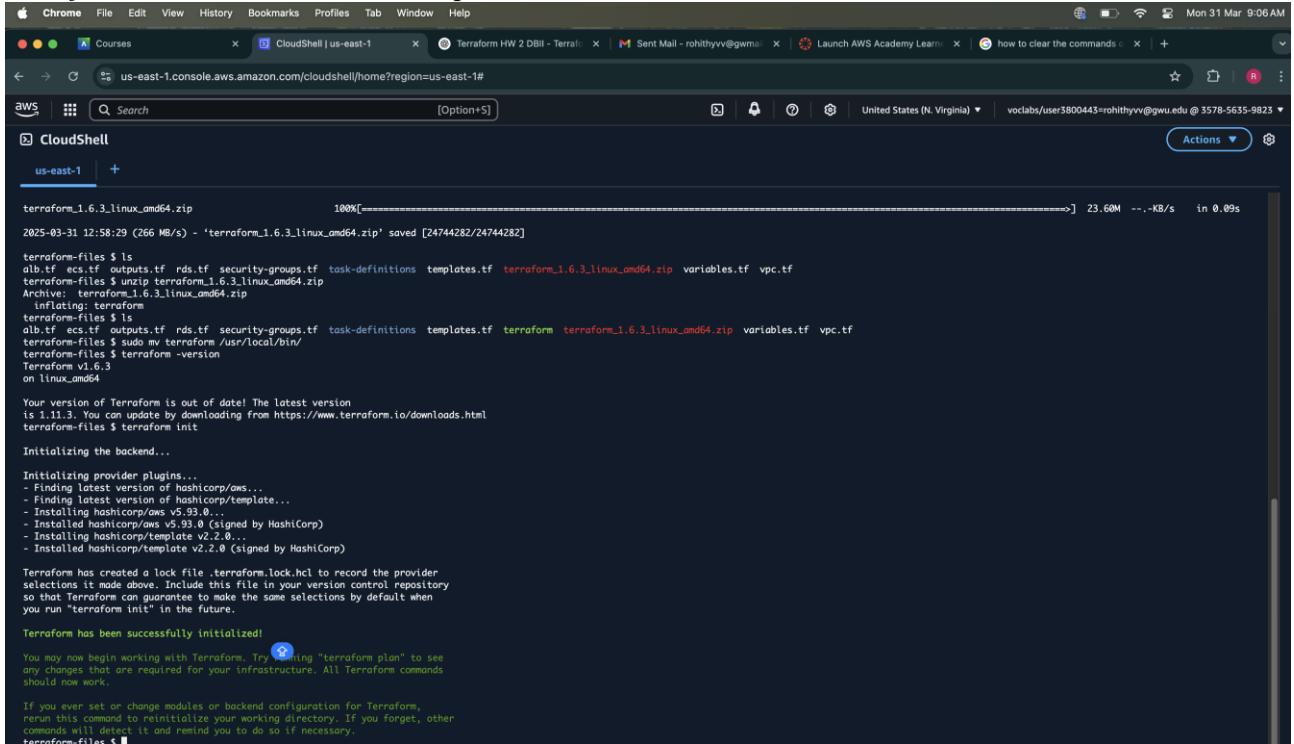
```
replace _MACOSX_terraform-files? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
inflating: _MACOSX_terraform-files
inflating: terraform-files/ecs.tf
inflating: _MACOSX_terraform-files/_ecs.tf
inflating: terraform-files/outputs.tf
inflating: _MACOSX_terraform-files/_outputs.tf
inflating: terraform-files/_DS_Store
inflating: _MACOSX_terraform-files/_DS_Store
inflating: terraform-files/security-groups.tf
inflating: _MACOSX_terraform-files/_security-groups.tf
inflating: _MACOSX_terraform-files/_task-definitions
inflating: terraform-files/templates.tf
inflating: _MACOSX_terraform-files/_templates.tf
inflating: terraform-files/variables.tf
inflating: _MACOSX_terraform-files/_variables.tf
inflating: terraform-files/alb.tf
inflating: _MACOSX_terraform-files/_alb.tf
inflating: terraform-files/rds.tf
inflating: _MACOSX_terraform-files/_rds.tf
inflating: terraform-files/vpc.tf
inflating: _MACOSX_terraform-files/_vpc.tf
inflating: terraform-files/task-definitions/wordpress.json
inflating: _MACOSX_terraform-files/task-definitions/_wordpress.json
$ cd terraform-files
bash: cd: terraform-files: Not a directory
$ cd terraform-files
terraform-files $ wget https://releases.hashicorp.com/terraform/1.6.3/terraform_1.6.3.linux_amd64.zip
--2025-03-31 12:58:29 - https://releases.hashicorp.com/terraform/1.6.3/terraform_1.6.3.linux_amd64.zip
Retrying with https://releases.hashicorp.com/ (releases.hashicorp.com)... 3.171.85.65, 3.171.85.80, 3.171.85.107, ...
Connecting to releases.hashicorp.com (releases.hashicorp.com)3.171.85.65:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 24744282 (24M) [application/xzip]
Saving to: 'terraform_1.6.3.linux_amd64.zip'

terraform_1.6.3.linux_amd64.zip          100%[=====] 23.60M  --.-KB/s   in 0.09s
2025-03-31 12:58:29 (266 MB/s) - 'terraform_1.6.3.linux_amd64.zip' saved [24744282/24744282]

terraform-files $ ls
alb.tf ecs.tf outputs.tf rds.tf security-groups.tf task-definitions templates.tf terraform_1.6.3.linux_amd64.zip variables.tf vpc.tf
Archive: terraform_1.6.3.linux_amd64.zip
  inflating: terraform
  terraform-files $ ls
alb.tf ecs.tf outputs.tf rds.tf security-groups.tf task-definitions templates.tf terraform terraform_1.6.3.linux_amd64.zip variables.tf vpc.tf
  terraform-files $
```

**Note:** Same as above, showing the terraform-files.zip uploaded into CloudShell and prepared for unzip.

*Fig 2.3: Navigate inside the unzipped folder to download the terraform\_1.6.3\_linux\_amd64.zip*



The screenshot shows a CloudShell terminal window within a web browser. The terminal is running on an AWS CloudShell instance in the us-east-1 region. The user has run the command `wget https://releases.hashicorp.com/terraform/1.6.3/terraform_1.6.3_linux_amd64.zip`. The output shows the file being downloaded from the HashiCorp releases page. The progress bar indicates the download is at 100% completion. The terminal also displays the Terraform initialization process, including provider plugin installations and the creation of a `.terraform.lock.hcl` file.

```

$ wget https://releases.hashicorp.com/terraform/1.6.3/terraform_1.6.3_linux_amd64.zip
--2025-03-31 12:58:29 (266 MB/s) - 'terraform_1.6.3_linux_amd64.zip' saved [24744282/24744282]

$ terraform init
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Finding latest version of hashicorp/template...
- Installing hashicorp/aws v5.93.0...
- Installed hashicorp/aws v5.93.0 (signed by HashiCorp)
- Installing hashicorp/template v2.2.0...
- Installed hashicorp/template v2.2.0 (signed by HashiCorp)

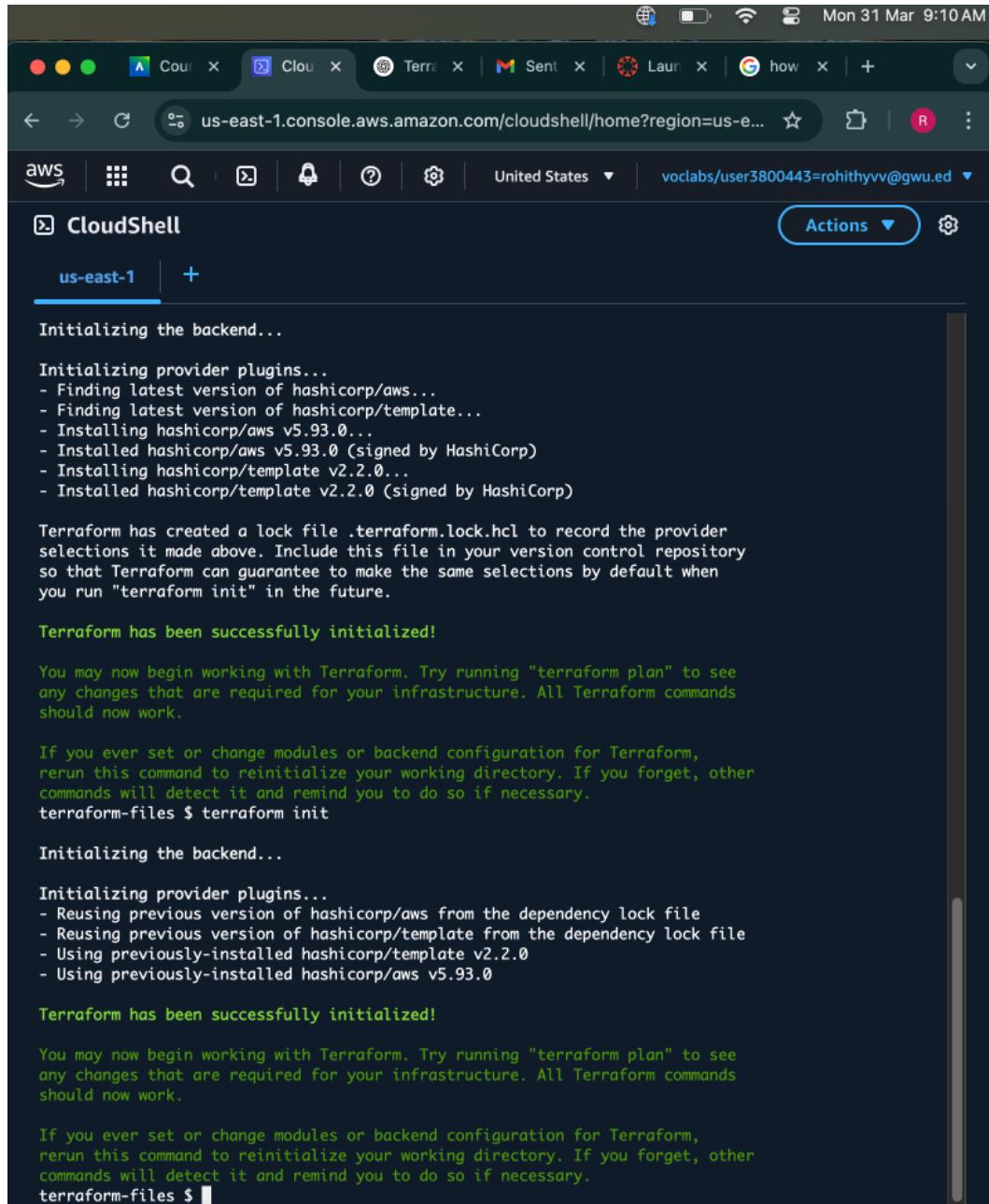
Terraform has created a lock file, .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

```

**Note:** After unzipping the Terraform files, navigate inside the folder and run:  
`wget https://releases.hashicorp.com/terraform/1.6.3/terraform_1.6.3_linux_amd64.zip`  
 to download the required binary, then unzip it.

**Title:** Fig 2.4: Move the terraform file to root and initialize terraform



The screenshot shows a terminal window titled "CloudShell" in an AWS CloudShell interface. The session is set to "us-east-1". The terminal displays the following output from the "terraform init" command:

```
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Finding latest version of hashicorp/template...
- Installing hashicorp/aws v5.93.0...
- Installed hashicorp/aws v5.93.0 (signed by HashiCorp)
- Installing hashicorp/template v2.2.0...
- Installed hashicorp/template v2.2.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.

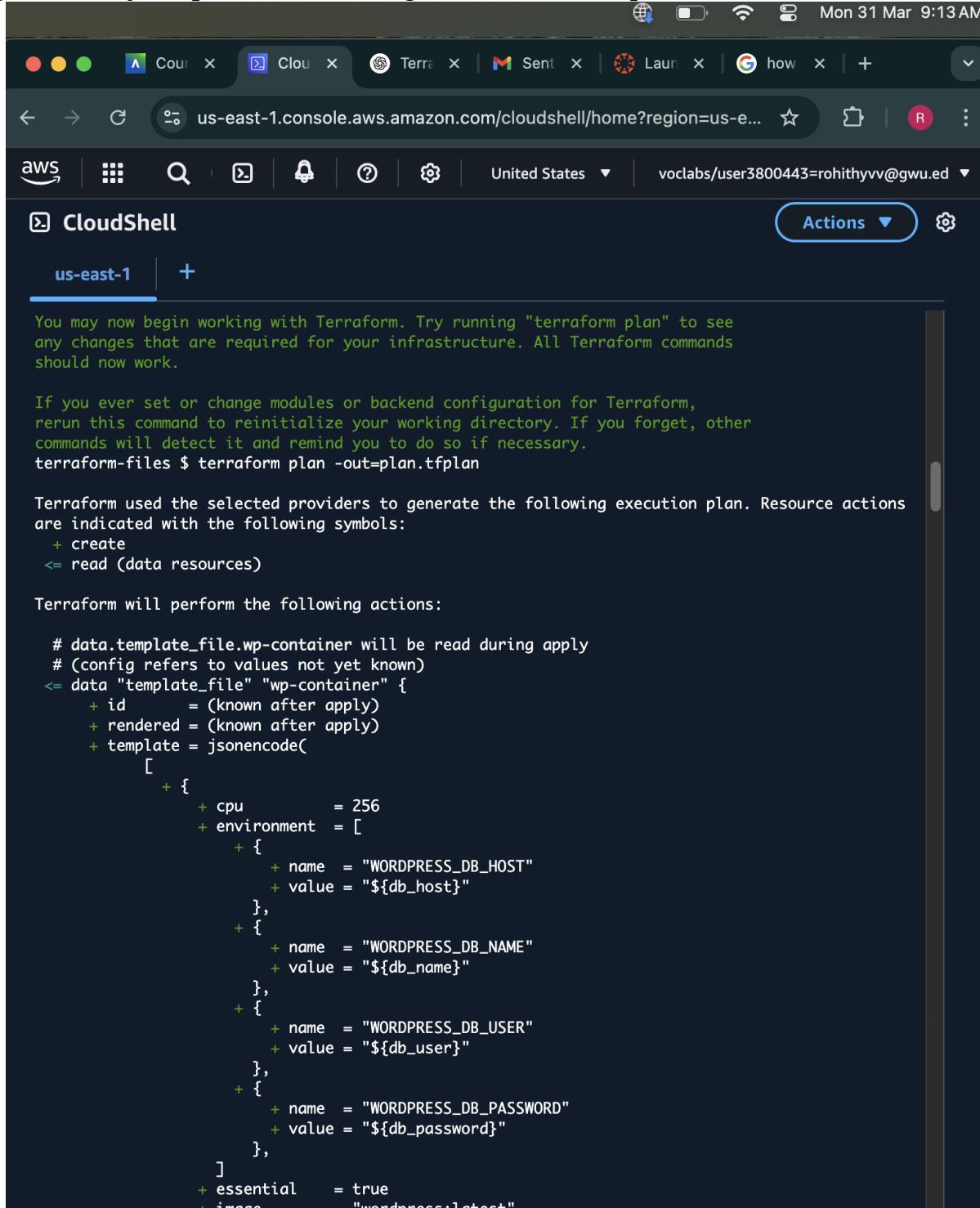
If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
terraform>
```

Below the terminal window, there is a status bar with the text "terraform>".

**Note:** Move the Terraform binary to /usr/local/bin to execute it globally. Then initialize with terraform init and prepare execution using terraform plan -out=plan.tfplan.

## plan.tfplan generation

*Fig 2.5: Terraform plan command to generate execution plan*



The screenshot shows a CloudShell terminal window in a web browser. The terminal is running on the 'us-east-1' region. The output of the 'terraform plan' command is displayed, showing the actions required to create a WordPress container. The terminal interface includes tabs for CloudWatch Metrics, CloudWatch Logs, Terraform, and others, along with AWS navigation links.

```

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.
terraform-files $ terraform plan -out=plan.tfplan

Terraform used the selected providers to generate the following execution plan. Resource actions
are indicated with the following symbols:
+ create
<= read (data resources)

Terraform will perform the following actions:

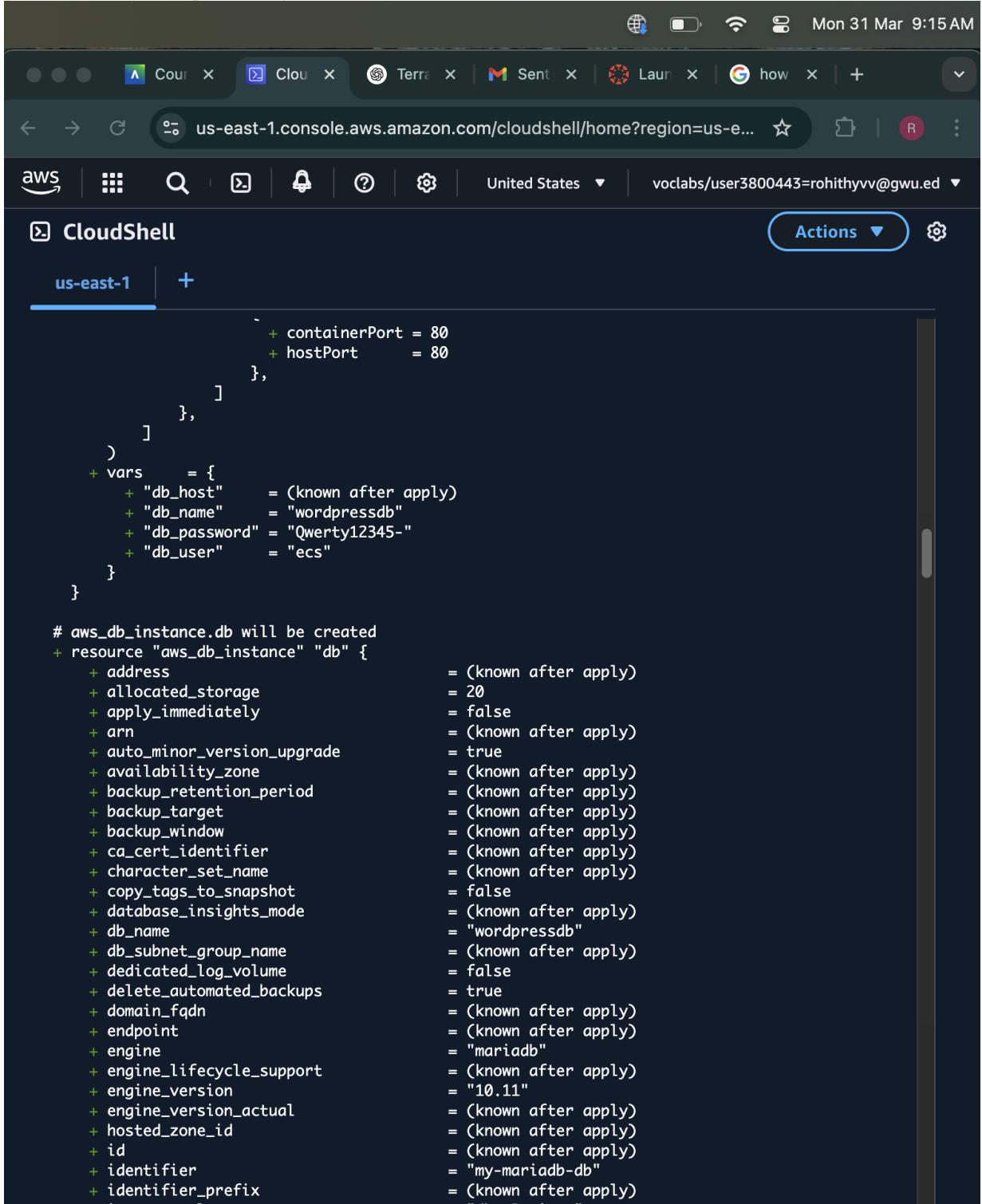
# data.template_file.wp-container will be read during apply
# (config refers to values not yet known)
<= data "template_file" "wp-container" {
    + id      = (known after apply)
    + rendered = (known after apply)
    + template = jsonencode(
        [
            + {
                + cpu          = 256
                + environment = [
                    + {
                        + name  = "WORDPRESS_DB_HOST"
                        + value = "${db_host}"
                    },
                    + {
                        + name  = "WORDPRESS_DB_NAME"
                        + value = "${db_name}"
                    },
                    + {
                        + name  = "WORDPRESS_DB_USER"
                        + value = "${db_user}"
                    },
                    + {
                        + name  = "WORDPRESS_DB_PASSWORD"
                        + value = "${db_password}"
                    },
                ],
            ]
        )
    + essential  = true
    - image      = "wordpress:latest"
}

```

**Note:** This command `terraform plan -out=plan.tfplan` analyzes the Terraform configuration and prepares an execution plan to create resources. It shows that a `template_file` data source is being rendered for the WordPress container with environment variables like `WORDPRESS_DB_HOST`, `DB_NAME`, `DB_USER`, and `DB_PASSWORD` which will be injected into the ECS task during deployment.

## RDS Instance Creation Plan

Fig 2.6: Terraform planning to create aws\_db\_instance resource



The screenshot shows a terminal window in the AWS CloudShell interface. The title bar indicates it's running on 'us-east-1' in the United States. The terminal content displays Terraform configuration code:

```

        + containerPort = 80
        + hostPort      = 80
    },
],
},
]
)
+
vars  = {
    + "db_host"      = (known after apply)
    + "db_name"      = "wordpressdb"
    + "db_password"  = "Qwerty12345-"
    + "db_user"      = "ecs"
}
}

# aws_db_instance.db will be created
+ resource "aws_db_instance" "db" {
    + address                      = (known after apply)
    + allocated_storage             = 20
    + apply_immediately            = false
    + arn                           = (known after apply)
    + auto_minor_version_upgrade   = true
    + availability_zone             = (known after apply)
    + backup_retention_period      = (known after apply)
    + backup_target                 = (known after apply)
    + backup_window                 = (known after apply)
    + ca_cert_identifier            = (known after apply)
    + character_set_name            = (known after apply)
    + copy_tags_to_snapshot          = false
    + database_insights_mode       = (known after apply)
    + db_name                       = "wordpressdb"
    + db_subnet_group_name          = (known after apply)
    + dedicated_log_volume          = false
    + delete_automated_backups     = true
    + domain_fqdn                  = (known after apply)
    + endpoint                      = (known after apply)
    + engine                         = "mariadb"
    + engine_lifecycle_support      = (known after apply)
    + engine_version                = "10.11"
    + engine_version_actual         = (known after apply)
    + hosted_zone_id                = (known after apply)
    + id                            = "my-mariadb-db"
    + identifier_prefix              = (known after apply)
}

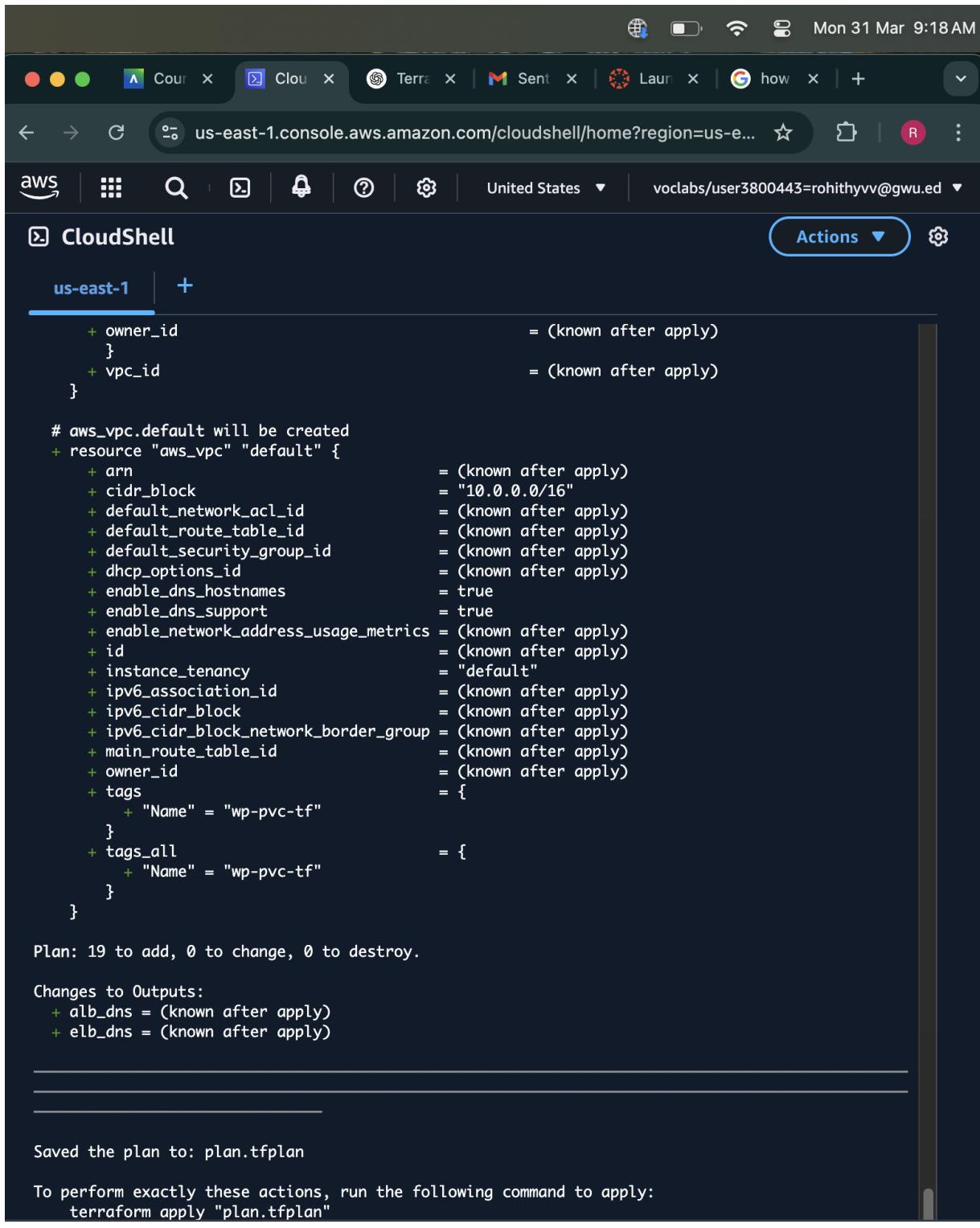
```

**Note:** This screenshot shows part of the execution plan where Terraform prepares to create an aws\_db\_instance resource for the WordPress database. It defines the engine as mariadb,

with a database name wordpressdb, username ecs, and password, and specifies instance class db.t3.micro. This RDS instance will be provisioned as part of the infrastructure.

## VPC Resource Creation Plan

Fig 2.7: Terraform planning to create aws\_vpc resource



```

aws_vpc.default will be created
+ resource "aws_vpc" "default" {
    + arn                               = (known after apply)
    + cidr_block                         = "10.0.0.0/16"
    + default_network_acl_id            = (known after apply)
    + default_route_table_id           = (known after apply)
    + default_security_group_id        = (known after apply)
    + dhcp_options_id                  = (known after apply)
    + enable_dns_hostnames             = true
    + enable_dns_support               = true
    + enable_network_address_usage_metrics = (known after apply)
    + id                                = (known after apply)
    + instance_tenancy                 = "default"
    + ipv6_association_id              = (known after apply)
    + ipv6_cidr_block                  = (known after apply)
    + ipv6_cidr_block_network_border_group = (known after apply)
    + main_route_table_id              = (known after apply)
    + owner_id                          = (known after apply)
    + tags                             = {
        + "Name" = "wp-pvc-tf"
    }
    + tags_all                         = {
        + "Name" = "wp-pvc-tf"
    }
}

Plan: 19 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ alb_dns = (known after apply)
+ elb_dns = (known after apply)

Saved the plan to: plan.tfplan

To perform exactly these actions, run the following command to apply:
  terraform apply "plan.tfplan"

```

No  
te:  
Thi  
s  
scr  
een  
sho  
t  
sho  
ws  
Ter  
raf  
or  
m's  
pla  
n  
to  
cre  
ate  
a  
cus  
to  
m  
VP  
C  
na  
me  
d  
wp  
-  
pv  
c-tf  
wit  
h  
CI  
DR  
blo  
ck  
10.

0.0.0/16, DNS support, and hostname enabled. The plan also confirms that 19 resources will be added with no changes or deletions, and that outputs such as alb\_dns and elb\_dns will be generated after the apply step.

**Title:** Fig 2.8: Final command to implement terraform is apply

CloudShell | us-east-1 | us-east-1.console.aws.amazon.com/cloudshell/home?region=us-east-1#

aws\_db\_instance.db: Still creating... [2m10s elapsed]  
aws\_lb.default: Still creating... [2m30s elapsed]  
aws\_lb.default: Still creating... [2m30s elapsed]  
aws\_lb.default: Still creating... [2m40s elapsed]  
aws\_lb.default: Still creating... [2m50s elapsed]  
aws\_lb.default: Still creating... [2m50s elapsed]  
aws\_lb.default: Still creating... [3m0s elapsed]  
aws\_lb.default: Still creating... [3m0s elapsed]  
aws\_lb.default: Still creating... [3m10s elapsed]  
aws\_lb.default: Still creating... [3m10s elapsed]  
aws\_lb.instance.db: Still creating... [3m10s elapsed]  
aws\_lb.instance.db: Still creating... [3m10s elapsed]  
aws\_lb.listener.default: Creation complete after 3m12s [id=arn:aws:elasticloadbalancing:us-east-1:357856359823:loadbalancer/app/wp-alb-tf/15de6752ef23e721]  
aws\_lb.listener.default: Creating...  
aws\_lb.listener.default: Creation complete after is [id=arn:aws:elasticloadbalancing:us-east-1:357856359823:listener/app/wp-alb-tf/15de6752ef23e721/d8beb45d3b7127d]  
aws\_db\_instance.db: Still creating... [3m20s elapsed]  
aws\_db\_instance.db: Still creating... [3m30s elapsed]  
aws\_db\_instance.db: Still creating... [3m40s elapsed]  
aws\_db\_instance.db: Still creating... [3m50s elapsed]  
aws\_db\_instance.db: Still creating... [4m0s elapsed]  
aws\_db\_instance.db: Still creating... [4m10s elapsed]  
aws\_db\_instance.db: Still creating... [4m20s elapsed]  
aws\_db\_instance.db: Still creating... [4m30s elapsed]  
aws\_db\_instance.db: Still creating... [4m40s elapsed]  
aws\_db\_instance.db: Still creating... [4m50s elapsed]  
aws\_db\_instance.db: Still creating... [5m0s elapsed]  
aws\_db\_instance.db: Still creating... [5m10s elapsed]  
aws\_db\_instance.db: Still creating... [5m20s elapsed]  
aws\_db\_instance.db: Still creating... [5m30s elapsed]  
aws\_db\_instance.db: Creation complete after 5m35s [id=db-TYL2W4BSUCBJL5NOKXWH3XXI]  
data.template\_file.wp-container: Reading...  
data.template\_file.wp-container: Read complete after 0s [id=4485c8e434d7c87be2e6a0bfe81a2ba5c3b308071b92277fa97f4fcfc952376]  
aws\_ecs\_task\_definition.task: Creating...  
aws\_ecs\_task\_definition.task: Creation complete after 0s [id=wp-task]  
aws\_ecs\_service.service: Creating...  
aws\_ecs\_service.service: Creation complete after 1s [id=arn:aws:ecs:us-east-1:357856359823:service/ecs-wordpress/wp-service]

Apply complete! Resources: 19 added, 0 changed, 0 destroyed.

Outputs:

```
alb_dns = "wp-alb-tf-482608135.us-east-1.elb.amazonaws.com"
elb_dns = "wp-alb-tf-482608135.us-east-1.elb.amazonaws.com"
terraform_files = ""
```

**Note:** Apply the execution plan using `terraform apply plan.tfplan`. This command deploys all resources including VPC, ECS, ALB, and RDS.

**Title:** Fig 2.9: WordPress installation page via Load Balancer

```

aws Chrome File Edit View History Bookmarks Profiles Tab Window Help
Courses X Terraform X Task con... X CloudSh... X VPC | us-... X VPC | us-... X New Tab X Sent Mail X Launch X how to cl... X wp-alb-tf X WordPress X Mon 31 Mar 9:41 AM
us-east-1.console.aws.amazon.com/cloudshell/home?region=us-east-1#
aws CloudShell [Option+S] Actions
CloudShell
us-east-1 +
aws_db_instance.db: Still creating... [2m10s elapsed]
aws_lb.default: Still creating... [2m20s elapsed]
aws_db_instance.db: Still creating... [2m20s elapsed]
aws_lb.default: Still creating... [2m20s elapsed]
aws_db_instance.db: Still creating... [2m30s elapsed]
aws_lb.default: Still creating... [2m40s elapsed]
aws_db_instance.db: Still creating... [2m40s elapsed]
aws_lb.default: Still creating... [2m50s elapsed]
aws_db_instance.db: Still creating... [2m50s elapsed]
aws_lb.default: Still creating... [2m50s elapsed]
aws_db_instance.db: Still creating... [3m0s elapsed]
aws_lb.default: Still creating... [3m10s elapsed]
aws_db_instance.db: Still creating... [3m10s elapsed]
aws_lb.default: Creation complete after 3m12s [id=arn:aws:elasticloadbalancing:us-east-1:357856359823:loadbalancer/app/wp-alb-tf/15de6752ef23e721]
aws_db_instance.db: Still creating... [3m20s elapsed]
aws_lb.listener.default: Creation complete after 1s [id=arn:aws:elasticloadbalancing:us-east-1:357856359823:listener/app/wp-alb-tf/15de6752ef23e721/d8beb45d3b712d]
aws_db_instance.db: Still creating... [3m20s elapsed]
aws_db_instance.db: Still creating... [3m30s elapsed]
aws_db_instance.db: Still creating... [3m40s elapsed]
aws_db_instance.db: Still creating... [3m50s elapsed]
aws_db_instance.db: Still creating... [4m0s elapsed]
aws_db_instance.db: Still creating... [4m10s elapsed]
aws_db_instance.db: Still creating... [4m20s elapsed]
aws_db_instance.db: Still creating... [4m30s elapsed]
aws_db_instance.db: Still creating... [4m40s elapsed]
aws_db_instance.db: Still creating... [4m50s elapsed]
aws_db_instance.db: Still creating... [5m0s elapsed]
aws_db_instance.db: Still creating... [5m10s elapsed]
aws_db_instance.db: Still creating... [5m20s elapsed]
aws_db_instance.db: Creation complete after 5m35s [id=db-TYLC2W4BSUC8J7L5N0XKVH3XXI]
data.template_file.wp-container: Reading
data.template_file.wp-container: Read complete after 8s [id=4485c8e434d7c87be2e6a0fe81a2b05dc3b308071b92277fa97f4fcfc952376]
aws_ecs_task_definition.task: Creating...
aws_ecs_task_definition.task: Creation complete after 0s [id=wp-task]
aws_ecs_service.service: Creating...
aws_ecs_service.service: Creation complete after 0s [id=arn:aws:ecs:us-east-1:357856359823:service/ecs-wordpress/wp-service]

Apply complete! Resources: 19 added, 0 changed, 0 destroyed.

Outputs:
elb_dns = "wp-alb-tf-482608135.us-east-1.elb.amazonaws.com"
elb_dns = "wp-alb-tf-482608135.us-east-1.elb.amazonaws.com"
terraform_files $ curl http://wp-alb-tf-482608135.us-east-1.elb.amazonaws.com
terraform_files $

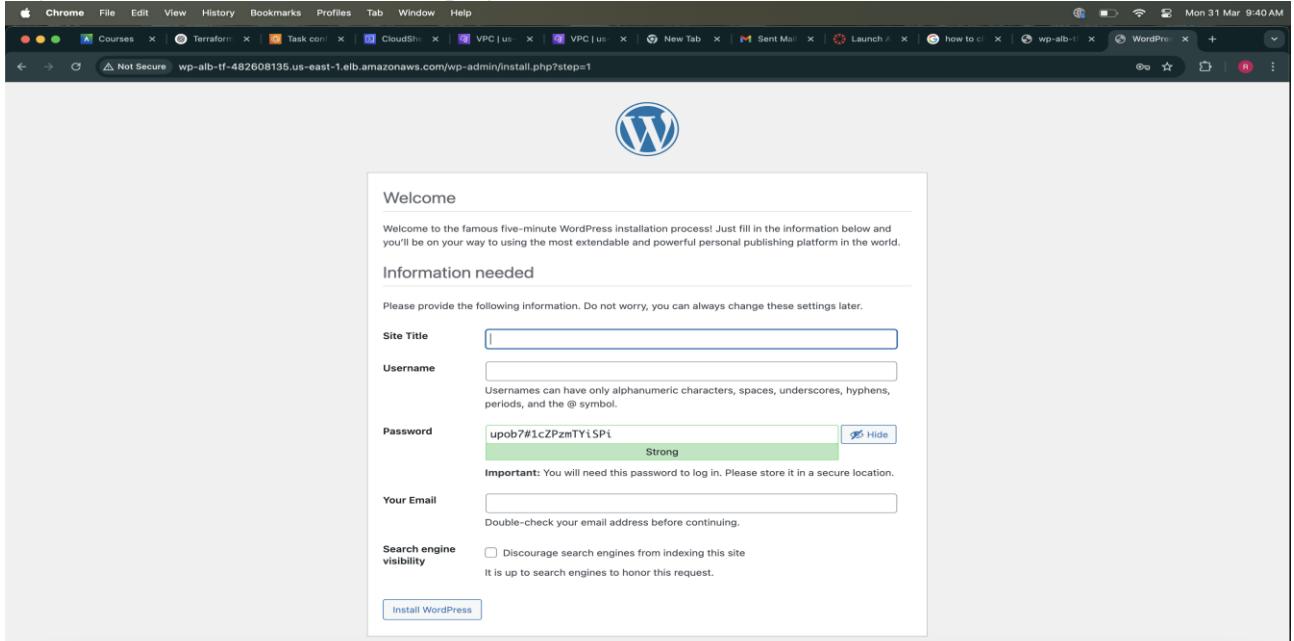
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

```

**Note:** Access the public ALB URL to open the WordPress installation page. This confirms the ECS container is running the WordPress service.

## Part 3: WordPress Setup Page

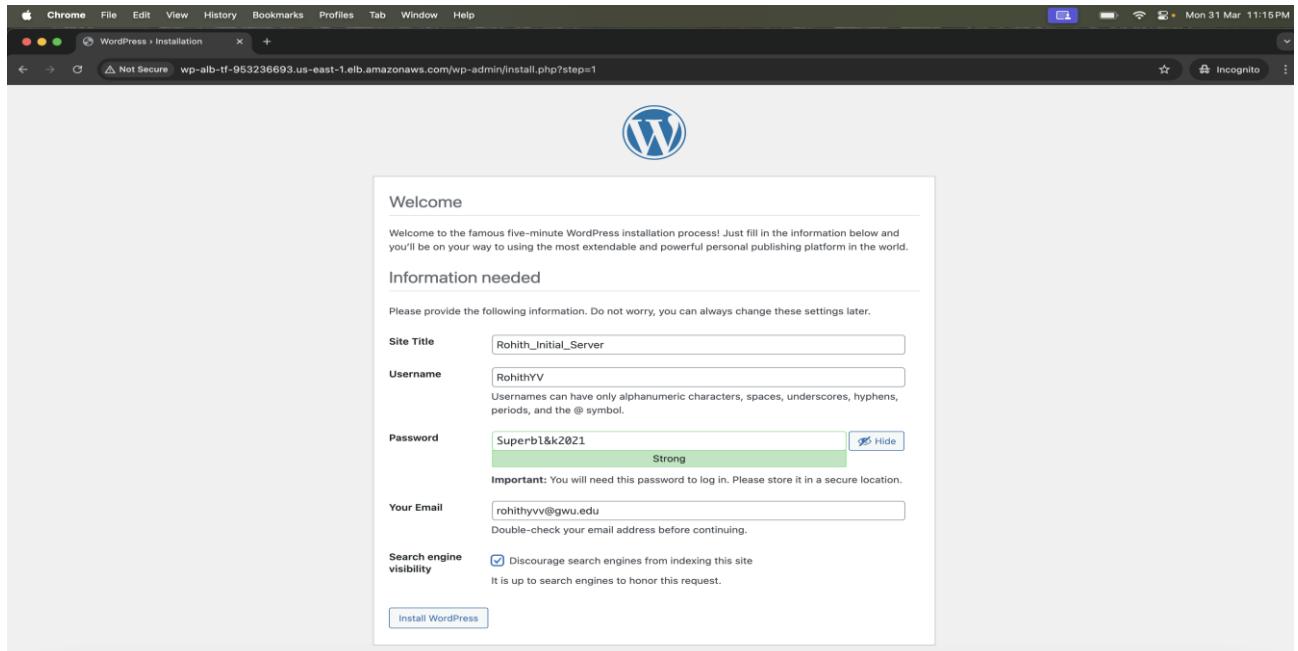
Fig 3.1: WordPress installation page loaded via ALB



**Note:** This screenshot shows the default WordPress setup wizard accessed through the Application Load Balancer (ALB) URL. It confirms that the ECS Fargate task running the WordPress container is active and accessible publicly. The form prompts for site title, username, password, and email to complete the WordPress installation process.

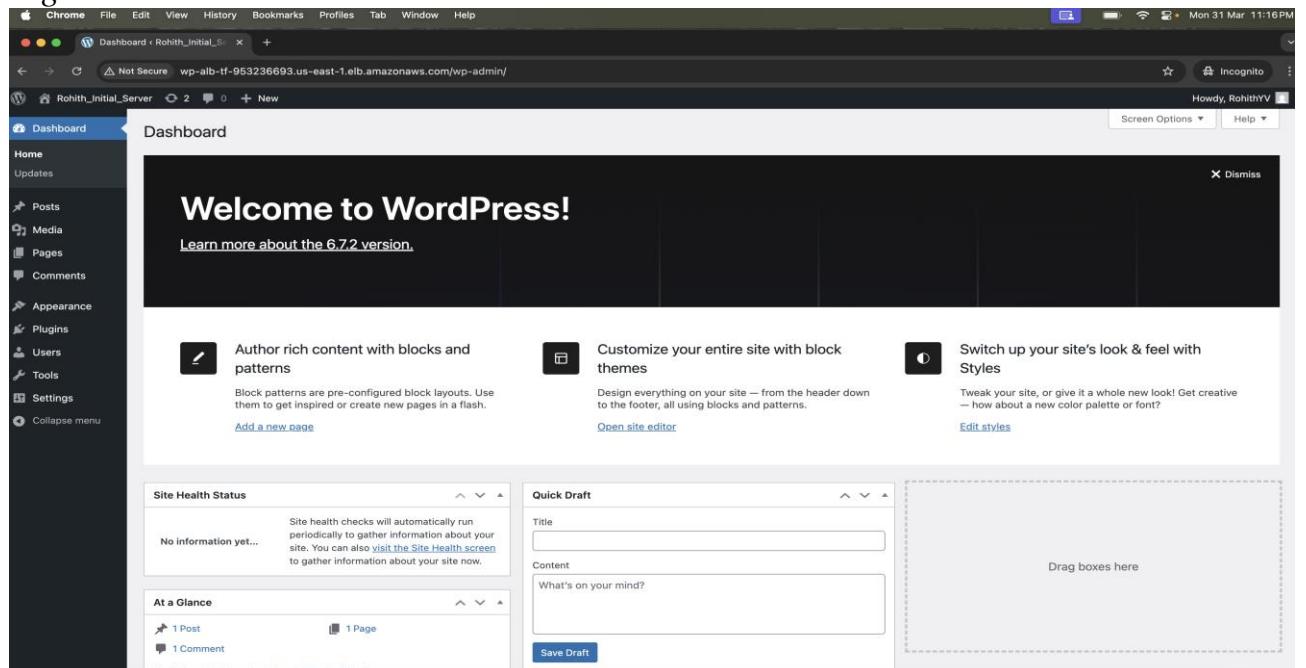
Fig 3.2: WordPress Initial Setup Screen on AWS ECS Fargate

## Homework 2 Deploying a Scalable WordPress Infrastructure on AWS Using Terraform RYV



**Note:** This screenshot shows the successful deployment of WordPress on AWS using ECS Fargate, RDS, and Application Load Balancer. The instance is accessible through the ALB DNS, and the WordPress installation page is now live. The user is prompted to set up the site title, admin username, password, and email as part of the famous 5-minute WordPress setup process.

Fig 3.3: WordPress Admin Dashboard on AWS



**Note:** This screenshot confirms successful deployment and configuration of WordPress on AWS using ECS Fargate, RDS, and an Application Load Balancer. The admin dashboard is accessible via the ALB endpoint, and the WordPress version 6.7.2 is fully operational. The user can now manage content, pages, plugins, and themes directly from the dashboard interface.

Fig 3.4: WordPress Activity Log Viewer on AWS Deployment

## Homework 2 Deploying a Scalable WordPress Infrastructure on AWS Using Terraform RYV

The screenshot shows the 'Activity Log Viewer' page of the WP Activity Log plugin. The left sidebar includes links for Dashboard, Log viewer, Notifications, Settings, Enable/Disable Events, Help & Contact Us, Posts, Media, Pages, Comments, Appearance, Plugins (with 1 update), Users, Tools, Settings, and a Collapse menu. The main area displays a table of log entries with columns: Bulk actions, ID, Severity, Date, User, IP, Object, Event Type, and Message. There are 7 items listed:

Bulk actions	ID	Severity	Date	User	IP	Object	Event Type	Message
<input type="checkbox"/>	6052	!	April 1, 2025 3:16:36.000 am	RohithYY Administrator	69.143.0.226	WP Activity Log	Modified	Changed the Activity log retention to Delete events older than 3 months. Previous setting: Keep all data
<input type="checkbox"/>	6067	!	April 1, 2025 3:16:36.000 am	RohithYY Administrator	69.143.0.226	Cron Jobs	Created	A new recurring task (cron job) called wsal_summary_weekly_report has been created. Task's first run : April 7, 2025 12:00:00.000 am Task's interval : Once Weekly
<input type="checkbox"/>	6067	!	April 1, 2025 3:16:36.000 am	RohithYY Administrator	69.143.0.226	Cron Jobs	Created	A new recurring task (cron job) called wsal_cleanup_hook has been created. Task's first run : April 1, 2025 3:16:36.000 am Task's interval : Once Hourly
<input type="checkbox"/>	6328	!	April 1, 2025 3:16:36.000 am	RohithYY Administrator	69.143.0.226	WP Activity Log	Modified	Modified the recipients of the Weekly Summary of Activity Log. New recipient: rohithyyv@gwu.edu Previous recipient:
<input type="checkbox"/>	6319	!	April 1, 2025 3:16:36.000 am	RohithYY Administrator	69.143.0.226	WP Activity Log	Enabled	Changed the status of the Weekly Summary of Activity Log..
<input type="checkbox"/>	6311	!	April 1, 2025 3:16:36.000 am	RohithYY Administrator	69.143.0.226	WP Activity Log	Modified	Modified the recipients of the Daily Summary of Activity Log. New recipient: rohithyyv@gwu.edu Previous recipient:
<input type="checkbox"/>	6310	!	April 1, 2025 3:16:36.000 am	RohithYY Administrator	69.143.0.226	WP Activity Log	Disabled	Changed the status of the Daily Summary of Activity Log..

**Note:** This displays the **WP Activity Log** plugin in action within the WordPress dashboard hosted on AWS ECS. It logs key administrative activities such as cron job creation, log retention changes, and summary email updates. This visibility into backend events helps ensure operational transparency and security for WordPress deployments, especially in production environments.

Fig 3.5: Publishing First WordPress Post – “Hi, I am Rohith”

The screenshot shows the 'Add New Post' screen in the WordPress dashboard. The title field contains 'Hi, I am Rohith'. The content area has a single paragraph: 'I would like to start wordpress'. The right sidebar shows the post status as 'Hi, I am Rohith' is now live. It also includes fields for 'POST ADDRESS' (http://wp-alb-tf-953236693.us-east-1.elb.amazonaws.com) and buttons for 'View Post' and 'Add New Post'.

**Note:** This shows a newly created and published blog post using the WordPress block editor on a site deployed through AWS ECS Fargate. The post, titled “*Hi, I am Rohith*”, confirms successful content creation and publishing functionality. The post is now publicly accessible via the ALB-provided DNS, verifying that the entire WordPress pipeline — from setup to live content — is working as expected.

Fig 3.6: WordPress Activity Log Showing Published Post and System Events

## Homework 2 Deploying a Scalable WordPress Infrastructure on AWS Using Terraform RYV

The screenshot shows a Chrome browser window displaying the 'Activity Log Viewer' for a WordPress site. The URL is `wp-alb-tf-953236693.us-east-1.elb.amazonaws.com/wp-admin/admin.php?page=wsal-auditlog`. The page title is 'Activity Log Viewer'. The left sidebar includes links for Dashboard, WP Activity Log (selected), Log viewer, Notifications, Settings, Enable/Disable Events, Help & Contact Us, and Upgrade to Premium. The main content area is titled 'Activity Log Viewer' and lists 9 items. The columns are: ID, Severity, Date, User, IP, Object, Event Type, and Message. The log entries include:

ID	Severity	Date	User	IP	Object	Event Type	Message
2001	⚠️	April 1, 2025 3:18:09.000 am	RohithVV Administrator	69.143.0.226	Post	Published	Published the post Hi, I am Rohith. Post ID: 6 Post type: post Post status: publish <a href="#">View the post in editor URL</a>
6066	ℹ️	April 1, 2025 3:18:09.000 am	RohithVV Administrator	69.143.0.226	Cron Jobs	Created	A new one-time task called do_pings has been scheduled. The task is scheduled to run on: April 1, 2025 3:18:09.000 am
6052	❗️	April 1, 2025 3:16:36.000 am	RohithVV Administrator	69.143.0.226	WP Activity Log	Modified	Changed the Activity log retention to Delete events older than 3 months. Previous setting: Keep all data
6067	ℹ️	April 1, 2025 3:16:36.000 am	RohithVV Administrator	69.143.0.226	Cron Jobs	Created	A new recurring task (cron job) called wsal_summary_weekly_report has been created. Task's first run: April 7, 2025 12:00:00.000 am Task's interval: Once Weekly
6067	ℹ️	April 1, 2025 3:16:36.000 am	RohithVV Administrator	69.143.0.226	Cron Jobs	Created	A new recurring task (cron job) called wsal_cleanup_hook has been created. Task's first run: April 1, 2025 3:16:36.000 am Task's interval: Once Hourly
6328	⚠️	April 1, 2025 3:16:36.000 am	RohithVV Administrator	69.143.0.226	WP Activity Log	Modified	Modified the recipients of the Weekly Summary of Activity Log. New recipient: rohithvv@gwu.edu Previous recipient:
6319	⚠️	April 1, 2025 3:16:36.000 am	RohithVV Administrator	69.143.0.226	WP Activity Log	Enabled	Changed the status of the Weekly Summary of Activity Log..
6311	⚠️	April 1, 2025 3:16:36.000 am	RohithVV Administrator	69.143.0.226	WP Activity Log	Modified	Modified the recipients of the Daily Summary of Activity Log.

Note: This activity log captures the successful publishing of the post “*Hi, I am Rohith*” along with related system events like scheduled cron jobs and configuration updates. The WP Activity Log plugin provides a detailed audit trail of administrator actions, ensuring transparency and traceability within the WordPress site hosted on AWS ECS Fargate.

**Title:** Fig 3.7: ECS Task Details

## Homework 2 Deploying a Scalable WordPress Infrastructure on AWS Using Terraform RYV

The screenshot shows the AWS CloudWatch Metrics interface. On the left, there's a sidebar with navigation links like 'CloudWatch Metrics', 'Metrics Insights', 'Metrics Data', 'Metrics Metrics', and 'Metrics Metrics Insights'. The main area displays a chart titled 'HelloWorld' with a single data series. The chart has two data points: one at approximately 1000 seconds with a value of 1, and another at approximately 1100 seconds with a value of 2. Below the chart, there's a table with columns 'Time' and 'Value'.

Time	Value
2023-03-31T10:00:00Z	1
2023-03-31T10:10:00Z	2

**Note:** The container is shown running inside ECS. The task is active, and the image wordpress:latest is deployed with essential variables.

### Title: Fig 3.8: ECS Cluster Overview

The screenshot shows the AWS CloudWatch Metrics interface. On the left, there's a sidebar with navigation links like 'CloudWatch Metrics', 'Metrics Insights', 'Metrics Data', 'Metrics Metrics', and 'Metrics Metrics Insights'. The main area displays a chart titled 'HelloWorld' with a single data series. The chart has two data points: one at approximately 1000 seconds with a value of 1, and another at approximately 1100 seconds with a value of 2. Below the chart, there's a table with columns 'Time' and 'Value'.

Time	Value
2023-03-31T10:00:00Z	1
2023-03-31T10:10:00Z	2

**Note:** Shows the ECS cluster ecs-wordpress is active with 1 running task associated with wp-service.

### Title: Fig 3.9: VPC and Subnet Structure

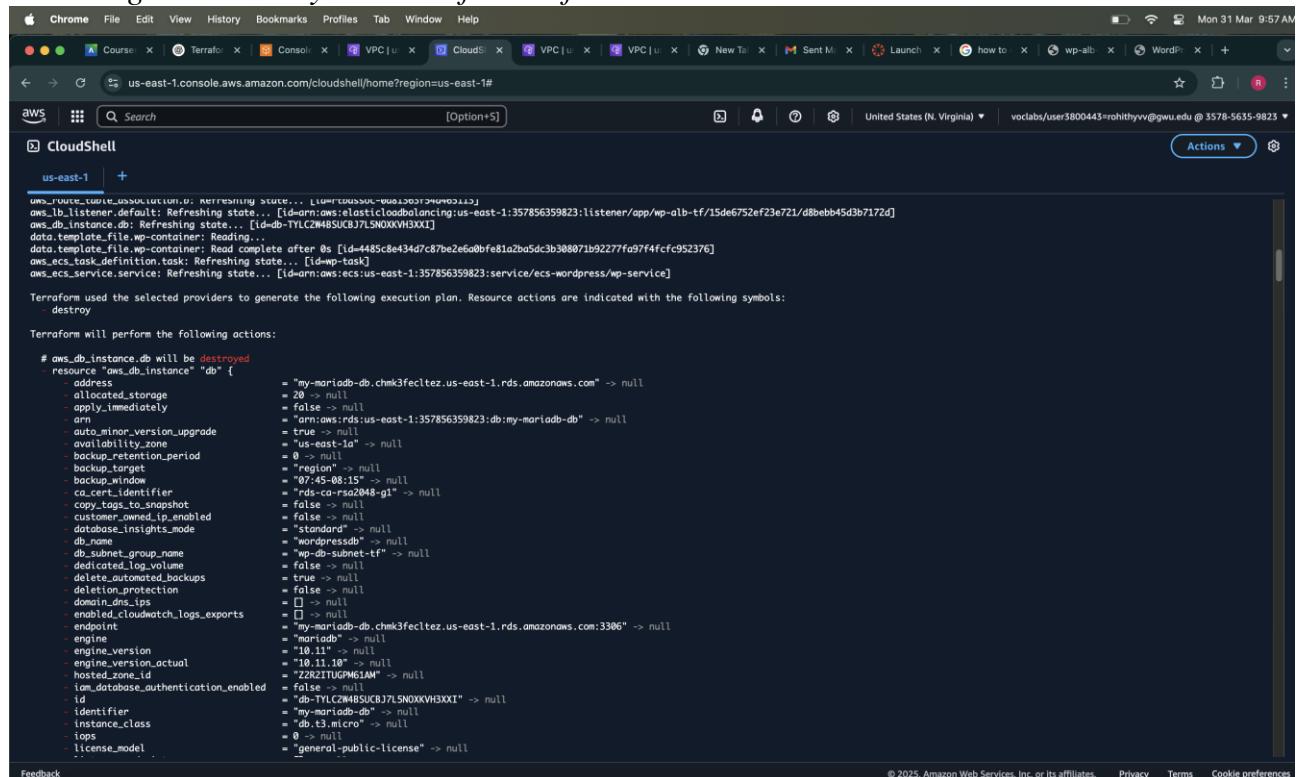
## Homework 2 Deploying a Scalable WordPress Infrastructure on AWS Using Terraform RYV

The screenshot shows the AWS CloudWatch Metrics console. The left sidebar navigation includes 'Clusters', 'Namespaces', 'Task definitions', 'Account settings', 'Install AWS Copilot', 'Amazon ECR', 'Repositories', 'AWS Batch', 'Documentation', 'Discover products', and 'Subscriptions'. The main content area displays the 'Cluster overview' for the 'ecs-wordpress' cluster. It shows the ARN (arn:aws:ecs:us-east-1:357856359823:cluster/ecs-wordpress), Status (Active), CloudWatch monitoring (Default), and Registered container instances (none). Below this, the 'Services' tab is selected, showing one service named 'wp-service' with an ARN of arn:aws:ecs:us-east-1:357856359823:service/ecs-wordpress/wp-service, Status Active, and Type REPLICA. A progress bar indicates 1/1 tasks running. The bottom of the page has a 'Tell us what you think' feedback link.

**Note:** Depicts the custom VPC wp-pvc-tf, with 3 public subnets and proper route tables, matching the Terraform network setup.

## Part – 4: Teardown Using Terraform Destroy

**Title:** Fig 4.1: Destroy the Terraform Infrastructure



```

aws_route_table_association: Refreshing state... [id=arn:aws:routing:us-east-1:357856359823:route-table/rtb-0000000000000000]
aws_lb_listener: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:357856359823:listener/app/wp-alb-tf/15de6752ef23e721/d8beb45d3b7172d]
aws_lb_listener: Refreshing state... [id=db-TYLZC948SUC8J7LSNOKXH3XXI]
data_template_file: wp contains: Reading...
data_template_file: wp contains: Read complete after 0s [id=4485c8e434d7c870e2e6a0bfe81a2b05d3b308071b92277fa97f4fcfc952376]
aws_ecs_task_definition: Refreshing state... [id=arn:aws:task]
aws_ecs_service: Refreshing state... [id=arn:aws:ecs:us-east-1:357856359823:service/ecs-wordpress/wp-service]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

# aws_db_instance.db will be destroyed
- resource "aws_db_instance" "db" {
    - address
    - allocated_storage
    - apply_immediately
    - auth_type
    - auto_minor_version_upgrade
    - availability_zone
    - backup_retention_period
    - backup_target
    - backup_window
    - ca_cert_identifier
    - character_set
    - customer_owned_ip_enabled
    - database_insights_mode
    - db_name
    - db_subnet_group_name
    - dedicated_log_volume
    - delete_automated_backups
    - deleted
    - deletion_protection
    - domain_dns_Ips
    - enabled_cloudwatch_logs_exports
    - endpoint
    - engine
    - engine_version
    - engine_version_actual
    - hosted_zone_id
    - iam_database_authentication_enabled
    - id
    - identifier
    - instance_class
    - ips
    - license_model
}

```

**Note:** After testing, terraform destroy is executed to tear down all resources to avoid charges. The log shows successful destruction of RDS, subnets, VPC, ALB, etc.

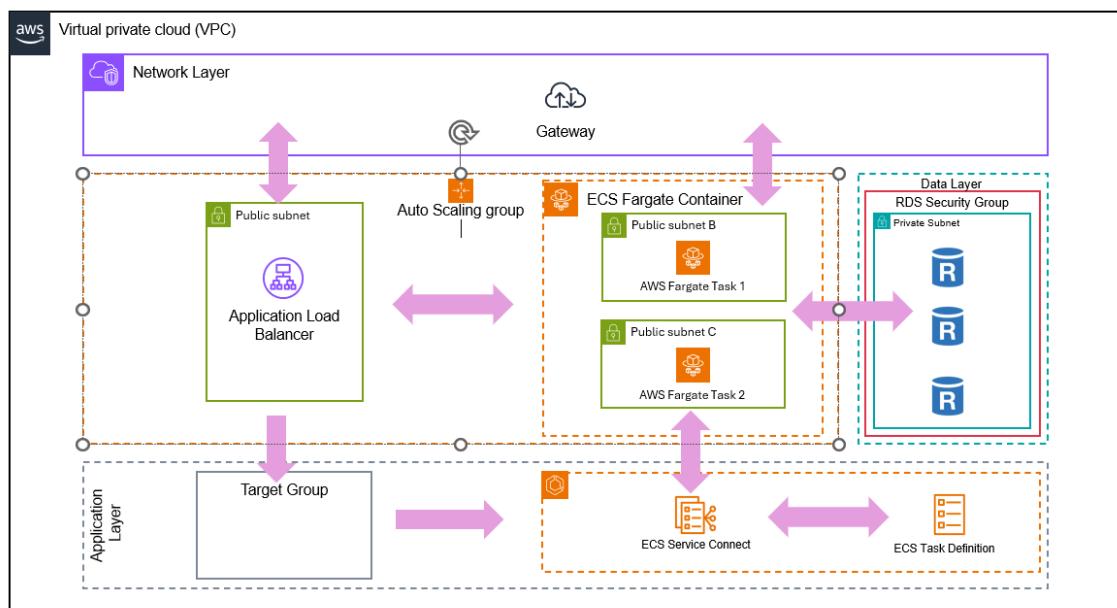
### Key Notes:

Once testing is completed, use the terraform destroy command to remove all AWS infrastructure components created during the deployment.

This avoids unnecessary AWS billing and ensures clean removal of all deployed services like ECS tasks, subnets, VPCs, and databases.

## Part – 5: Infrastructure Diagram

**Title:** Fig 5.1: Infrastructure Diagram



### Key Notes:

This architecture diagram represents a cloud-based application deployed on AWS using **ECS Fargate** for containerized workloads. It features an **Application Load Balancer (ALB)** distributing traffic across ECS tasks running in different public subnets, ensuring scalability and fault tolerance.

The **data layer** consists of an **RDS database** in a private subnet, protected by a security group, ensuring secure database access.