Version 2024-02-27

2110524 Cloud Computing Technologies Midterm Project "Infrastructure-as-Code"

You are going to automatically deploy a Wordpress website on Amazon web services using an infrastructure-as-code software called Terraform.

1. You should have a basic idea on how to setup wordpress from the previous activity
2. Download and setup Terraform on your computer or EC2 instance (https://www.terraform.io/)
3. In this project, you will create a terraform script that will deploy and provision
    a. A new VPC for your project. You may not use the default VPC in the region.
    b. An EC2 instance, with proper configurations, for installing and running the Wordpress app (https://wordpress.org/download/)
    c. An EC2 instance, with proper configurations, for installing and running a MariaDB database
    d. A new S3 bucket, with proper settings, and configure Wordpress to use it for media storage via WP Offload Media plugin (https://aws.amazon.com/th/blogs/compute/deploying-a-highly-available-wordpress-site-on-amazon-lightsail-part-2-using-amazon-s3-with-wordpress-to-securely-deliver-media-files/)
    e. Various other services/configurations (IAM User, VPC, Internet Gateway, security groups, etc.) necessary to make this a fully functioning Wordpress web app on AWS

Read these documents on how to set up Wordpress with S3.

1. https://aws.amazon.com/th/blogs/compute/deploying-a-highly-available-wordpress-site-on-amazon-lightsail-part-1-implementing-a-highly-available-lightsail-database-with-wordpress/
2. https://aws.amazon.com/th/blogs/compute/deploying-a-highly-available-wordpress-site-on-amazon-lightsail-part-2-using-amazon-s3-with-wordpress-to-securely-deliver-media-files/
3. https://deliciousbrains.com/wp-offload-media/doc/settings-constants/
4. https://deliciousbrains.com/wp-offload-media/doc/iam-roles/

Note : The goal of the project is to be able to automate deployment from scratch (source code), i.e., continuous deployment (CD). **You are not allowed to deploy Wordpress using any other means**. So, no docker image, Wordpress appliance, or snap package manager are allowed. Doing so would result in getting a score of 0 for this project. However, feel free to study those provisioning scripts for ideas. Additionally, you may want to set up Wordpress on an EC2 instance manually to see what you need to configure after installation. However, MariaDB can be installed via a package manager.

Additional requirements:

1. Your script shall accept the terraform.tfvars file and configure the region, availability_zone, bucket_name, etc. accordingly. (See : https://gist.github.com/theminer3746/150c52809e0235e99265ee1e252912cf)
2. For this project, we will use the **Ubuntu Server 22.04 LTS (HVM)** AMI. Because different regions use different AMI ID's for the same image, a single AMI ID can not

be used across regions. You will need to specify the AMI ID in your configuration file (see above).

3. You must use PHP version 8.3 and MariaDB version 10.6. This must be explicitly specified within your installation script.
   Do NOT use 3rd party repositories for PHP such as ppa:ondrej/php.
4. You should take security into consideration when creating security groups. Think of all the scenarios thoroughly. For example, the database instance should not be accessible by anyone on the Internet, and should only be accessible by the app instance's IP/subnet only. This translates to only allow incoming TCP connections to port 3306 (MariaDB's port) on the database instance from the app instance's IP/subnet only.
5. Note that your AWS keys should never be included in the Terraform script. Anyone who has set up the proper ~/.aws/credentials or roles should be able to run your script.
6. Make sure to add this ssh key to your ~/.ssh/authorized_keys to both Wordpress and MariaDB instances so I can connect to your wordpress installation ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIODaHqtrCOBpfD+meWggDG5gFEqnNDtpxnqQ7xWIfXfL cloud-wordpress

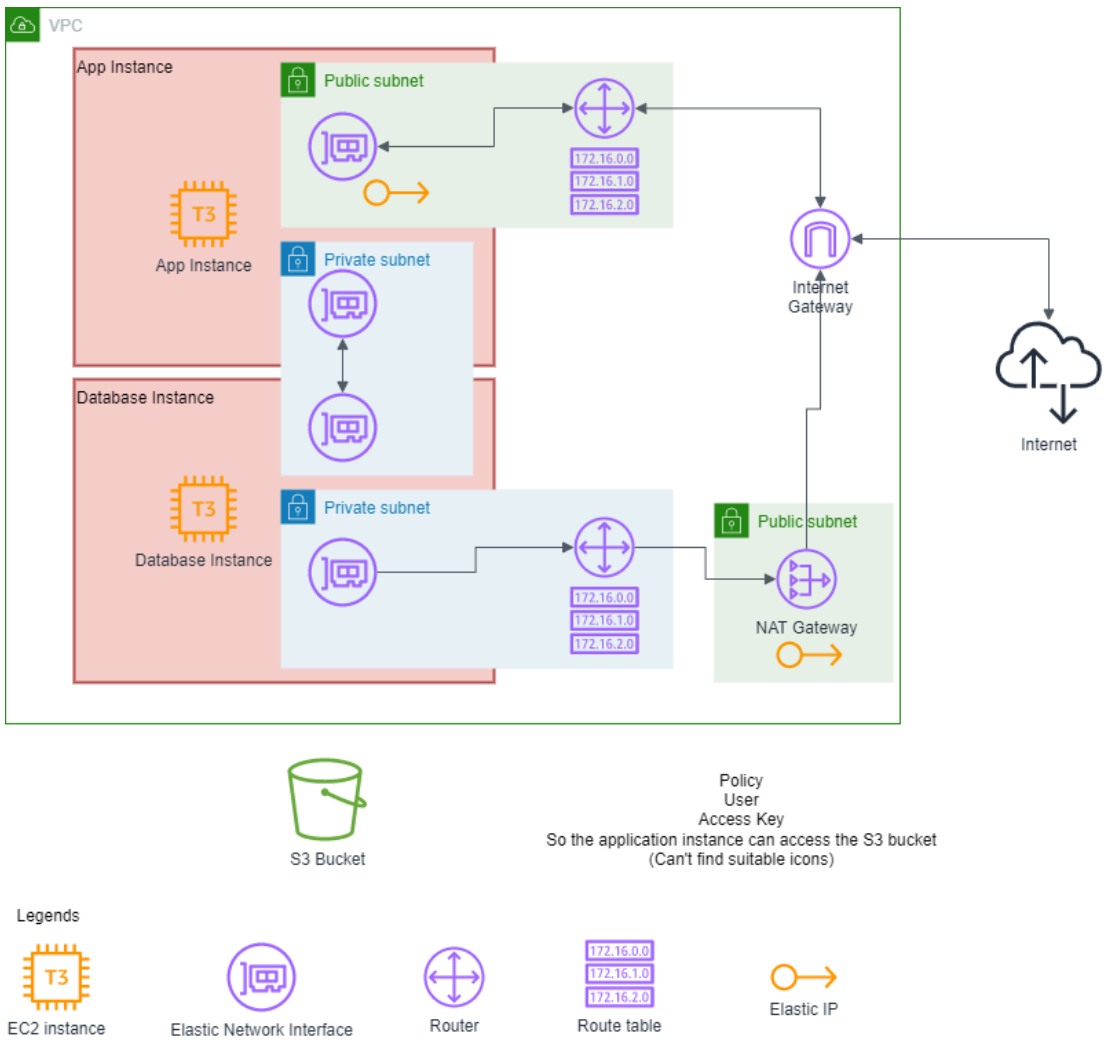


Figure 1: Your infrastructure should look like this after running terraform apply. Any unfamiliar services will require additional self-study.

Diagram explanation:

1. An EC2 instance (App) running Wordpress with 2 elastic network interfaces.
   a. Interface 1 has an elastic IP attached and is in a publicly accessible subnet associated with a routing table and an Internet gateway.
   Users will connect to Wordpress using the elastic IP on this interface.
   b. Interface 2 is in a private subnet used only to communicate with the database instance.
2. An EC2 instance (Database) running MariaDB, also with 2 elastic network interfaces.
   a. Interface 1 is in a private subnet with a route table associated with a NAT gateway which is associated with an elastic IP, so that your database instance can access the Internet to download MariaDB.
   Note : As good security practice, people must not be able to connect to the database instance from the Internet directly. Therefore, you should use a NAT gateway.
   b. Interface 2 is in a private subnet used only to communicate with the app instance.

Terraform resources that you may need to use (in no particular order) :

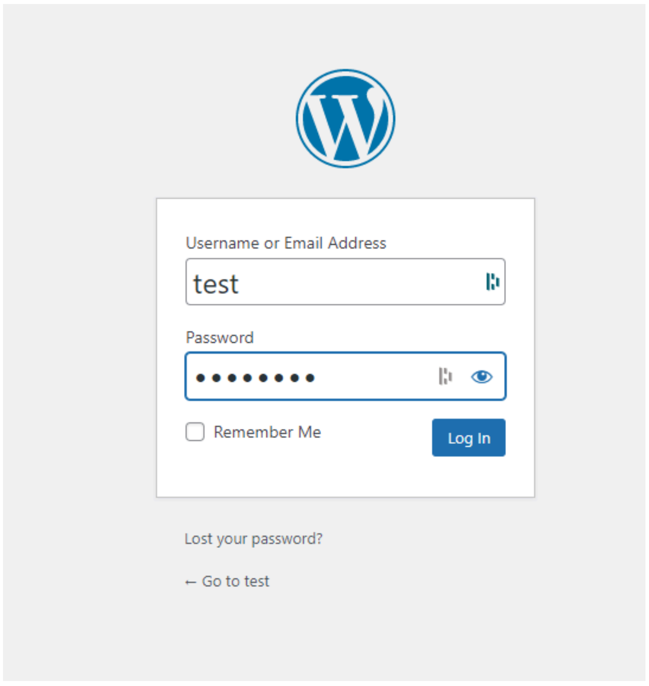
1. aws_vpc
2. aws_subnet
3. aws_network_interface
4. aws_security_group
5. aws_security_group_rule
6. aws_instance
   Note : You may use the user_data property to pass the application provisioning script to the instance (see https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance#user_data)
7. aws_internet_gateway
8. aws_nat_gateway
9. aws_route
10. aws_route_table
11. aws_route_table_association
12. aws_eip
13. aws_s3_bucket
14. aws_iam_access_key
15. aws_iam_user
16. aws_iam_policy

This is not a definitive list. You can use resources not in this list if you want.

Version 2024-02-27

Expected results

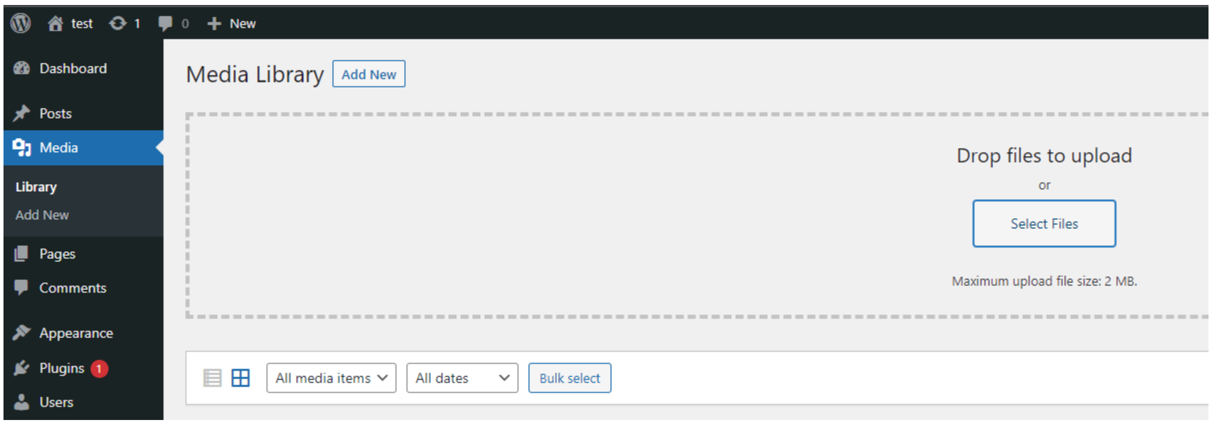
After running **terraform apply,** you should be able to login to the admin panel via http://<your-elastic-ip>/wp-admin


Username or Email Address

test

Password

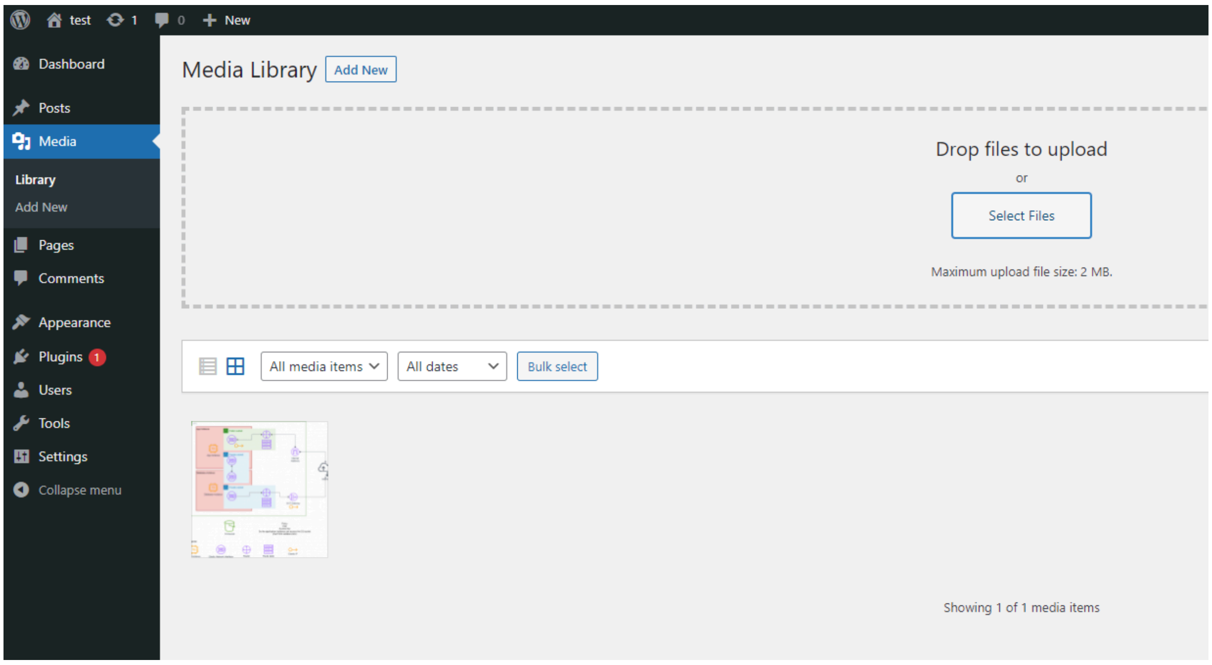
Remember Me



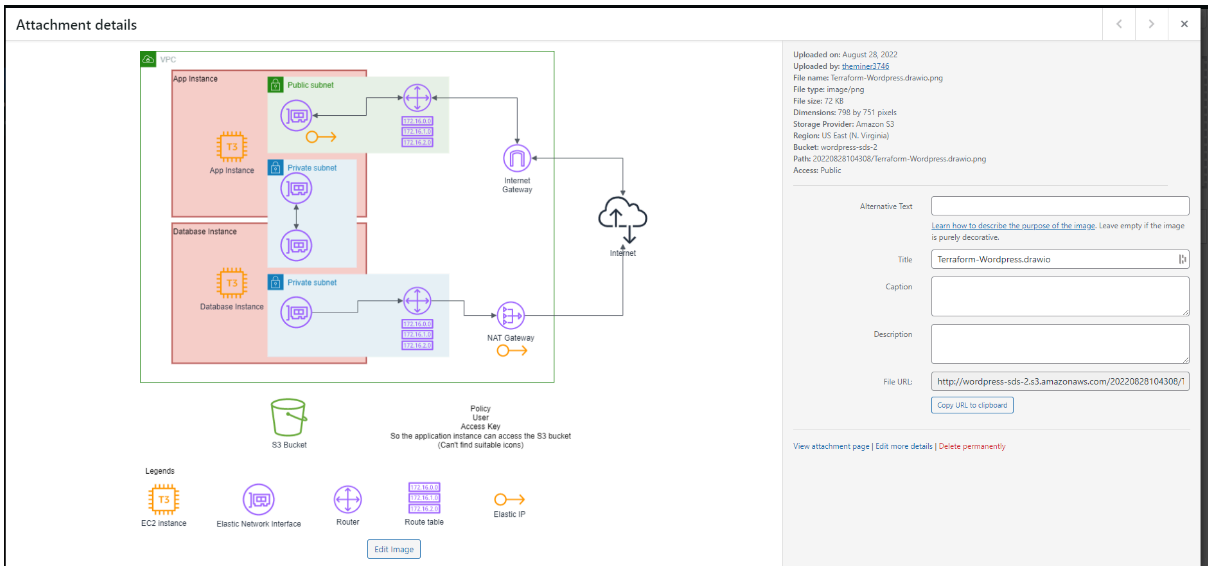
Maximum upload file size: 2 MB.

You should be able to upload media files to wordpress.

Version 2024-02-27



Click the uploaded media.



Uploaded on: August 28, 2022
Uploaded by: theminer3746
File name: Terraform-Wordpress.drawio.png
File type: image/png
File size: 72 KB
Dimensions: 798 by 751 pixels
Storage Provider: Amazon S3
Region: US East (N. Virginia)
Bucket: wordpress-sds-2
Path: 20220828104308/Terraform-Wordpress.drawio.png
Access: Public

Alternative Text	
	Learn how to describe the purpose of the image. Leave empty if the image is purely decorative.
Title	Terraform-Wordpress.drawio
Caption	
Description	
File URL:	http://wordpress-sds-2.s3.amazonaws.com/20220828104308/T

Copy URL to clipboard

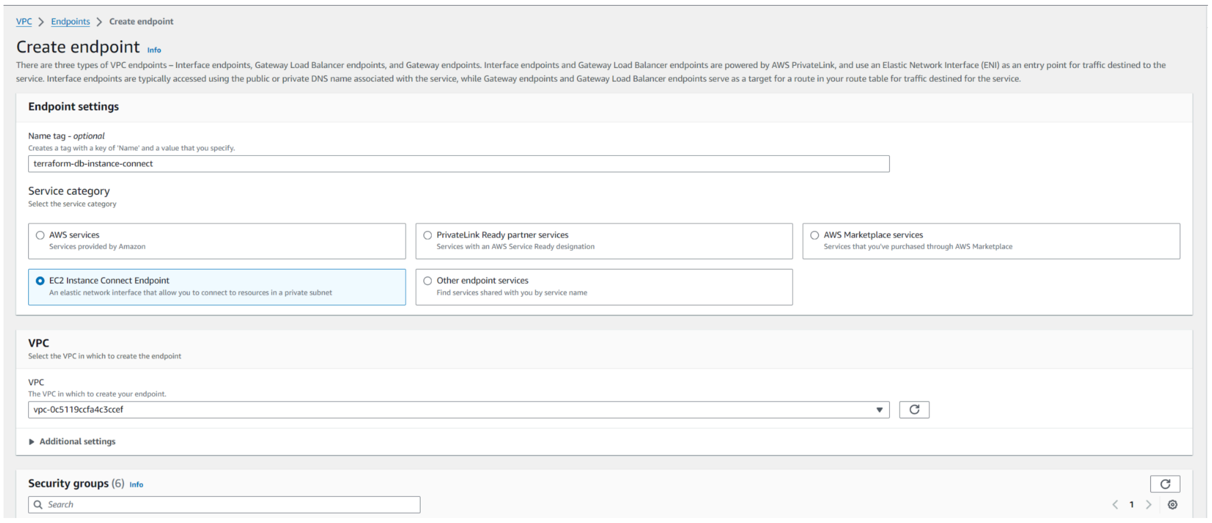
View attachment page | Edit more details | Delete permanently

Every file should have their url changed to s3 url and be publicly accessible such as http://wordpress-sds-2.s3.amazonaws.com/20220828104308/Terraform-Wordpress.drawio.png

Version 2024-02-27

Additional hints that may be useful

1. Take a look at cloudinit. It is a powerful provisioning tool. You can also use a plain bash script if you prefer.
2. If you use cloudinit, look at /var/log/cloud-init.log for execution results/errors.
3. Many tutorials use Apache as the web server. However, if you prefer Nginx, feel free to go ahead.
4. Since you can not directly ssh into the database instance because it is behind the NAT gateway, you can use a VPC endpoint in order to connect to the instance.

   VPC > Endpoints > Create endpoint

   Create endpoint Info

   There are three types of VPC endpoints – Interface endpoints, Gateway Load Balancer endpoints, and Gateway endpoints. Interface endpoints and Gateway Load Balancer endpoints are powered by AWS PrivateLink, and use an Elastic Network Interface (ENI) as an entry point for traffic destined to the service. Interface endpoints are typically accessed using the public or private DNS name associated with the service, while Gateway endpoints and Gateway Load Balancer endpoints serve as a target for a route in your route table for traffic destined for the service.

   Endpoint settings

   Name tag - *optional*
   Creates a tag with a key of 'Name' and a value that you specify.
   terraform-db-instance-connect

   Service category
   Select the service category

   - ○ AWS services — Services provided by Amazon
   - ○ PrivateLink Ready partner services — Services with an AWS Service Ready designation
   - ○ AWS Marketplace services — Services that you've purchased through AWS Marketplace
   - ● EC2 Instance Connect Endpoint — An elastic network interface that allow you to connect to resources in a private subnet
   - ○ Other endpoint services — Find services shared with you by service name

   VPC
   Select the VPC in which to create the endpoint

   VPC
   The VPC in which to create your endpoint.
   vpc-0c5119ccfa4c3ccef

   ▸ Additional settings

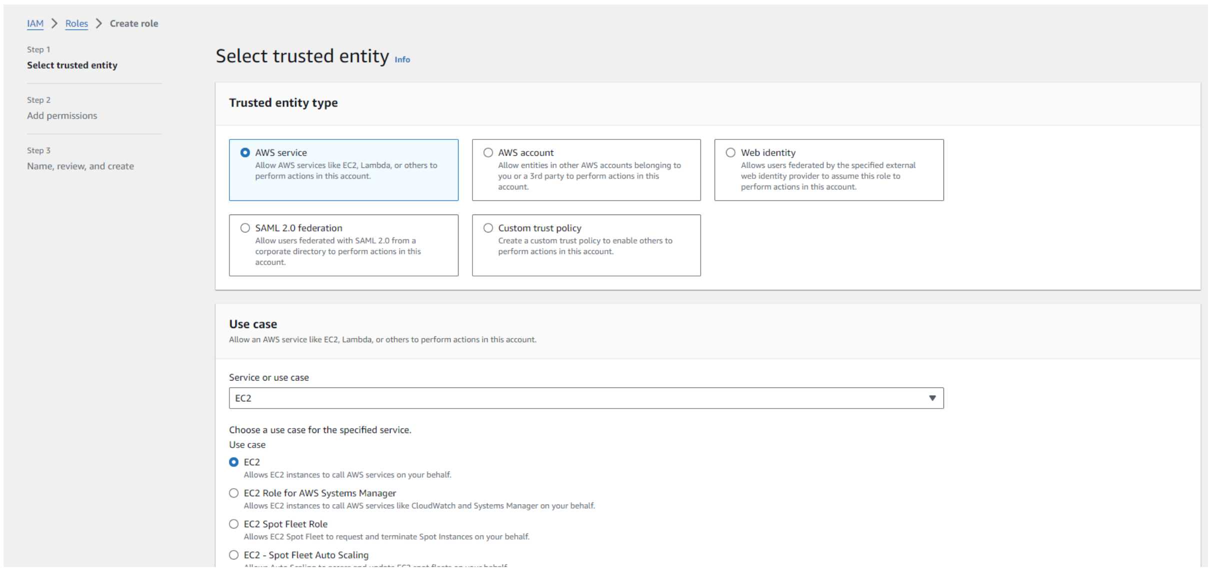
   Security groups (6) Info
   Search

   However, you should provision the VPC endpoint via Terraform or the terraform destroy command will not be able to delete the VPC/subnet automatically without deleting the VPC endpoint because of dependencies.
5. You can use `wp core install --url="${aws_eip.wordpress_ip.public_ip}" --admin_user="${var.admin_user}" --admin_password="${var.admin_pass}" --admin_email="exmaple@example.com" --title="Cloud" --skip-email` to configure the admin account from installation.

Version 2024-02-27

Setting up an EC2 instance to run terraform with suitable policy

1. Create an instance for running terraform provisioning. (This is not to be used for/by the web application directly.)
2. Add HashiCorp's signing key and repository to your package manager first. Then you can sudo apt install terraform. Follow the instructions here (https://www.hashicorp.com/official-packaging-guide)
3. Create an IAM role with AWS service trusted entity type and select EC2 as the use case.



IAM > Roles > Create role

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Select trusted entity Info

Trusted entity type

- [✓ AWS service] Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account: Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity: Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation: Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy: Create a custom trust policy to enable others to perform actions in this account.

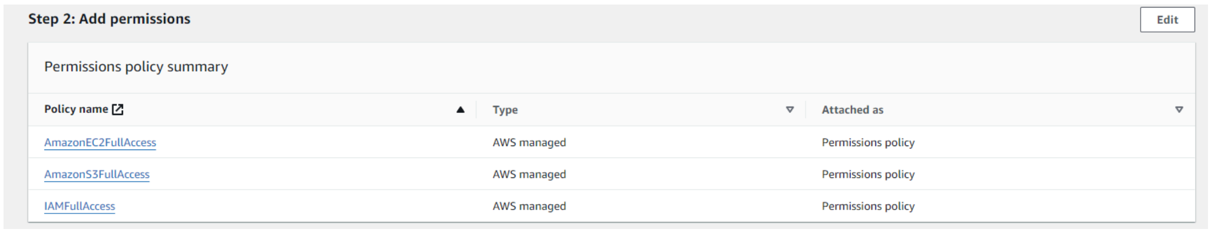
Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case: EC2

Choose a use case for the specified service.
Use case

- [✓ EC2] Allows EC2 instances to call AWS services on your behalf.
- EC2 Role for AWS Systems Manager: Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.
- EC2 Spot Fleet Role: Allows EC2 Spot Fleet to request and terminate Spot Instances on your behalf.
- EC2 - Spot Fleet Auto Scaling

4. In the Add permissions tab, select AmazonEC2FullAccess, AmazonS3FullAccess, and IAMFullAccess.

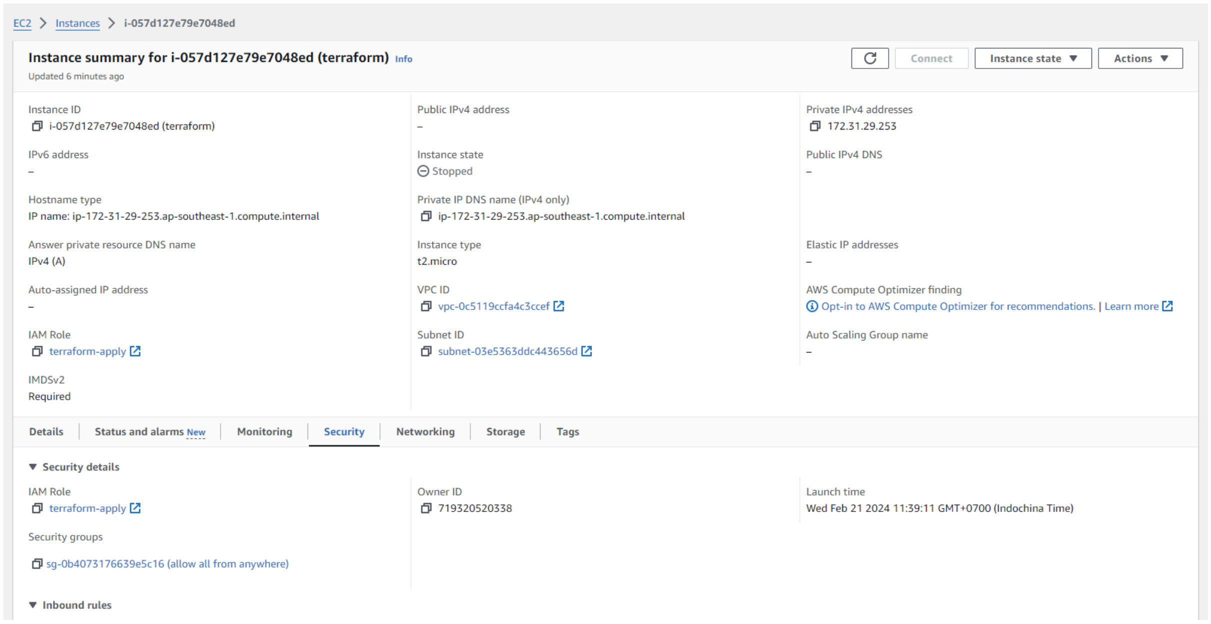


Step 2: Add permissions Edit

Permissions policy summary

Policy name	Type	Attached as
AmazonEC2FullAccess	AWS managed	Permissions policy
AmazonS3FullAccess	AWS managed	Permissions policy
IAMFullAccess	AWS managed	Permissions policy

5. On the terraform provisioning instance, attach the IAM role you created.



EC2 > Instances > i-057d127e79e7048ed

Instance summary for i-057d127e79e7048ed (terraform) Info
Updated 6 minutes ago

Instance ID: i-057d127e79e7048ed (terraform)	Public IPv4 address: –	Private IPv4 addresses: 172.31.29.253
IPv6 address: –	Instance state: Stopped	Public IPv4 DNS: –
Hostname type: IP name: ip-172-31-29-253.ap-southeast-1.compute.internal	Private IP DNS name (IPv4 only): ip-172-31-29-253.ap-southeast-1.compute.internal	
Answer private resource DNS name: IPv4 (A)	Instance type: t2.micro	Elastic IP addresses: –
Auto-assigned IP address: –	VPC ID: vpc-0c5119ccfa4c3ccef	AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role: terraform-apply	Subnet ID: subnet-03e5363ddc443656d	Auto Scaling Group name: –
IMDSv2: Required		

▼ Security details

IAM Role: terraform-apply	Owner ID: 719320520338	Launch time: Wed Feb 21 2024 11:39:11 GMT+0700 (Indochina Time)
Security groups: sg-0b4073176639e5c16 (allow all from anywhere)		

▼ Inbound rules

6. You can now use this instance to run terraform

Version 2024-02-27

What to submit by the deadline March 19th, 2024 @23:00:

1. Your terraform script.
2. A link to your youtube video (< 5 mins long) that
    a. Demo's your terraform apply
    b. Shows the instances and S3 bucket it provisions
    c. Demo's successful install of Wordpress and upload a file through the web UI
    d. Shows the objects in your S3 bucket after file upload

What to do on demo day March 20th, 2024 @13:00 (in class):

Instructors will assign you a region and you will change your terraform.tfvars file to configure your script to deploy resources in that region in your AWS account. You will run your terraform script with the new variables. After the script is done, you will demo the same content as your recorded video, and answer some questions about what you have done. Note that you should test deploying your script on other regions to confirm you understand the process and changes required prior to demo day.