AWS Use Case – 3

EFS file system for persistent container storage

**Amazon Elastic File System (EFS)** is a scalable, fully-managed cloud storage service designed to provide **shared, persistent storage** for Amazon ECS (Elastic Container Service). It allows containers to access and share the same file system, ensuring that data remains persistent even after the container stops or is replaced.

**Key Features of EFS for Container Storage:**

* **Scalability**: EFS automatically grows and shrinks as you add or remove files, allowing your storage to scale dynamically without provisioning.
* **Persistent Data**: EFS ensures that data is stored independently of the container's lifecycle. Even if a container is stopped or redeployed, the data remains available for future use.
* **Shared Access**: Multiple containers or ECS tasks can simultaneously access the same file system.
* **High Availability and Durability**: EFS stores data across multiple Availability Zones (AZs), providing high durability and availability by replicating data within a region.
* **Integration with ECS**: EFS can be easily integrated with Amazon ECS tasks. You can mount an EFS file system as a volume within your container, making it simple to manage persistent storage alongside stateless containers. This setup is common for applications like NGINX or Apache, where file storage for static assets (e.g., web pages, images) needs to persist beyond the container's lifetime.

**Use Cases:**

* **Web Hosting**: Storing static files such as HTML, CSS, images, and JavaScript files for NGINX or Apache web servers.
* **Content Management**: Applications like WordPress or Drupal where files need to persist even when containers are restarted.
* **Shared Workflows**: Collaborative applications where multiple containers need access to the same file system for processing or analysis.

**Hands On:**

Ref:https://docs.aws.amazon.com/AmazonECS/latest/developerguide/tutorial-efs-volumes.html

This configuration describes an Amazon ECS (Elastic Container Service) task definition that is designed to deploy a NGINX web server with an EFS (Elastic File System) volume mounted to it.

**1. Create an Amazon ECS cluster**

* Open the ECS console
* select the Region to use – us-east-1
* On the Clusters page, choose Create cluster.
* Under Cluster configuration, for Cluster name, enter “**EFS-tutorial”** for the cluster name.
* To change the VPC and subnets, under Networking, choose default VPC and then under Subnets, select each subnet.
* To add Amazon EC2 instances to your cluster, expand Infrastructure, and then select Amazon EC2 instances. Next, configure the Auto Scaling group which acts as the capacity provider:
  + To create an Auto Scaling group, from Auto Scaling group (ASG), select Create new group, and then provide the following details about the group:
    - For Operating system/Architecture, choose **Amazon Linux 2**.
    - For EC2 instance type, choose **t2.micro**.

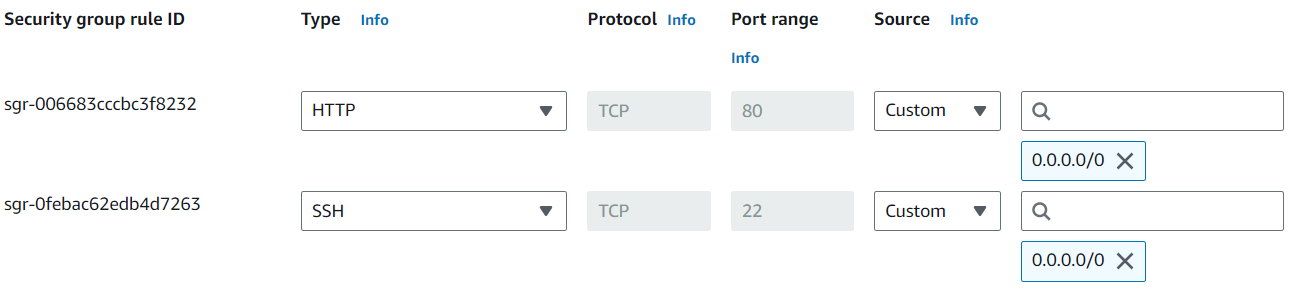
For SSH key pair, choose the pair that proves your identity when you connect to the instance.

* + - For Capacity, enter 1.
* Choose **Create**.

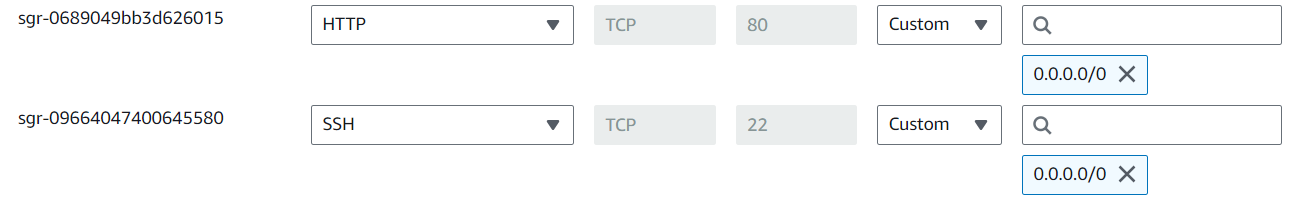
**2. Create a security group for Amazon EC2, ECS instances and the Amazon EFS file system**

you create a security group for your Amazon EC2 instances that allows inbound network traffic on port 80 and your Amazon EFS file system that allows inbound access from your container instances.

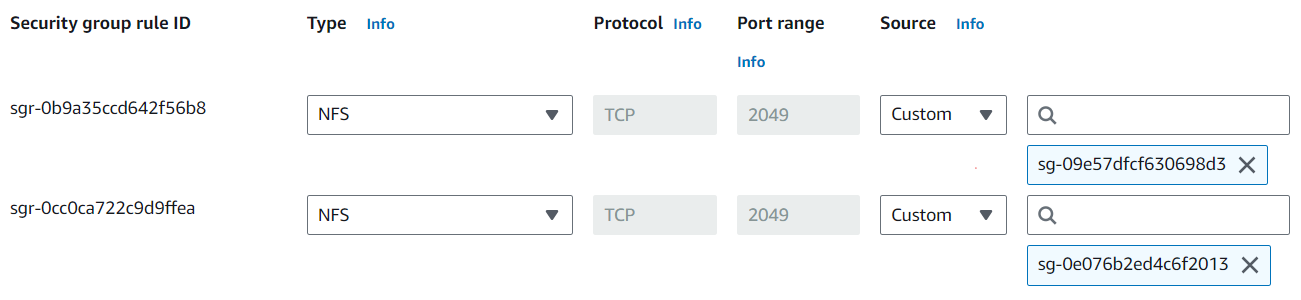
* EC2-Instance-SG



* ECS-Instance-SG



* EFS-SG



Allow NFS inbound from EC2 Instance SG and ECS Instance SG.

**3. Create an Amazon EFS file system**

* Open the EFS console - Create file system.
* Enter a name – **Central-EFS,** choose the VPC that your container instances are hosted.
* under Performance settings, choose the Bursting throughput
* (Optional) Enable encryption.
* On the Network access page, under Mount targets, replace the existing security group configuration for every availability zone with the security group you created for the file system in previous step.
* Review your file system options and choose Create to complete the process.
* From the File systems screen, note the **File system ID.**

**4. Add content to the Amazon EFS file system**

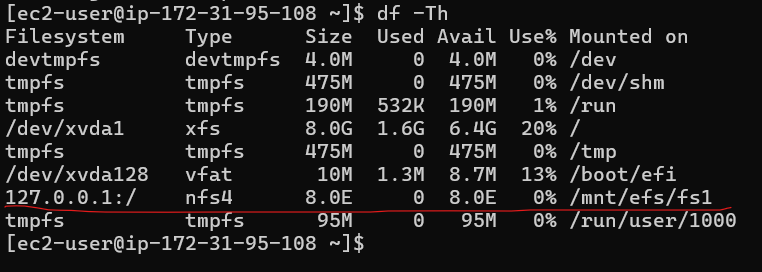
In this step, you mount the Amazon EFS file system to an Amazon EC2 instance and add content to it.

**Create an Amazon EC2 instance and mount the Amazon EFS file system,**

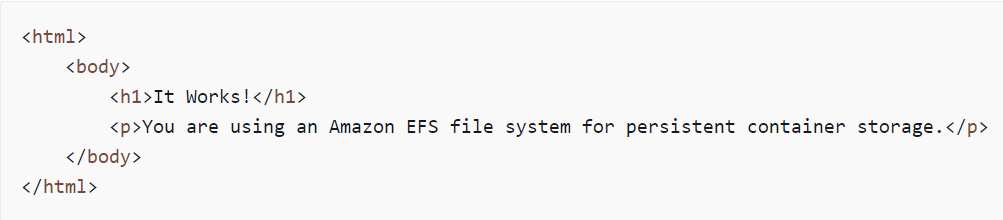
* Launch an EC2 Instance with below config.
* AMI: Amazon Linux 2 AMI
* Instance type: t2.micro.
* Select a key pair for SSH access to the instance.
* Under Network settings, select default VPC.
* Select a subnet and the instance security group- EC2-Instance-SG.
* Under **Configure storage**, choose the **Edit** button for file systems and then choose **EFS**. Select the file system you created in previous step.
* Mount point, leave the default value.
* Clear the **Automatically create and attach security groups**. Leave the other check box selected. Choose **Add shared file system**.
* Under **Advanced Details**, ensure that the user data script is populated automatically with the Amazon EFS file system mounting steps.

**Add content to the Amazon EFS file system**

* SSH to the Amazon EC2 instance.
* From the terminal window, run the **df -T** command to verify that the Amazon EFS file system is mounted.

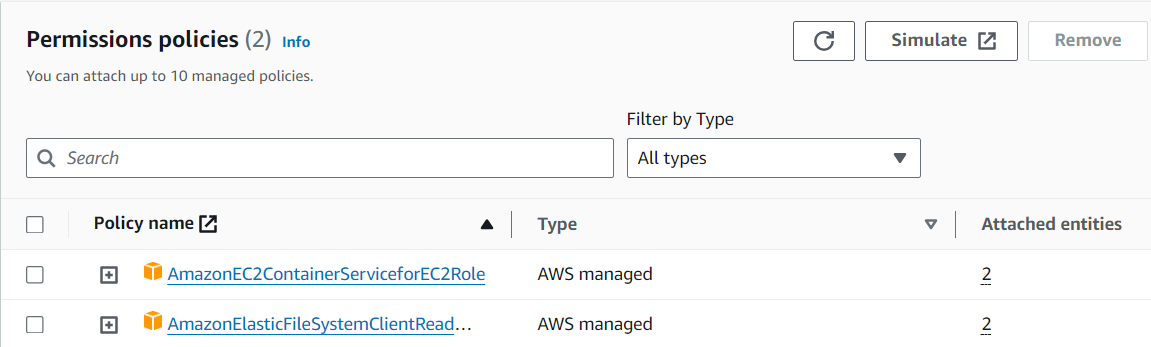


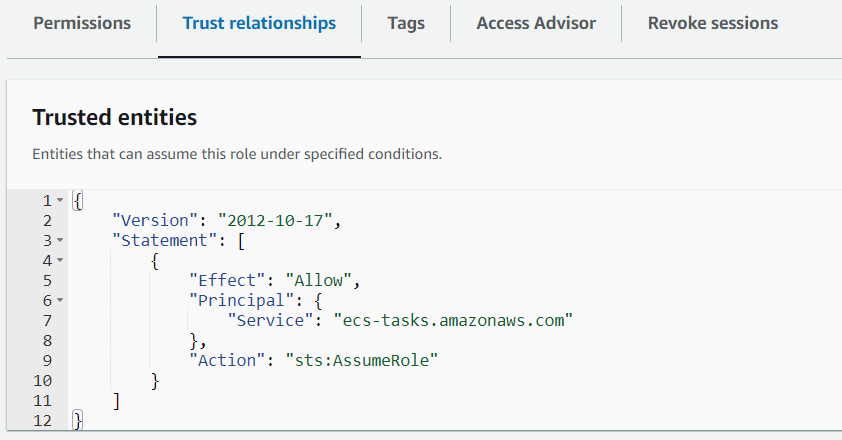
* Navigate to the directory that the Amazon EFS file system is mounted at. In the example above, that is /mnt/efs/fs1.
* Create a file named index.html with the following content:



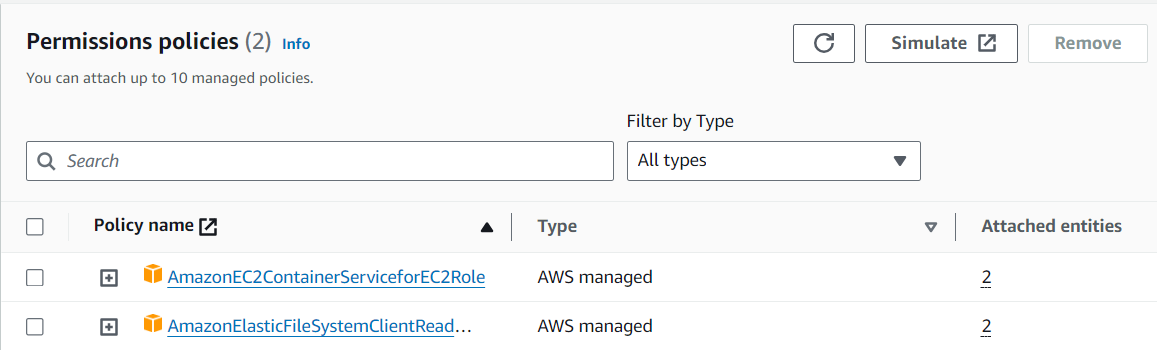
**5. Create IAM roles ecsTaskExecutionRole and ecsInstanceRole**

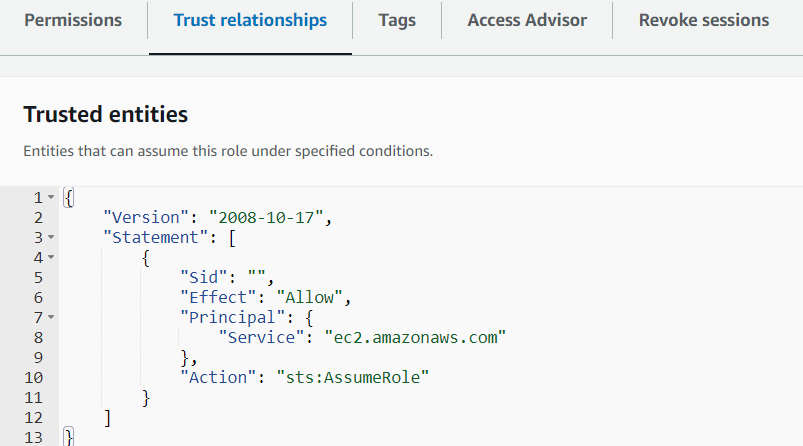
* **ecsTaskExecutionRole:**

****



* **ecsInstanceRole**

****

****

**6. Create a task definition**

The following task definition creates a data volume named efs-html. The nginx container mounts the host data volume at the NGINX root, /usr/share/nginx/html.

Create a new task definition using the Amazon ECS console.

* ECS console - choose **Task definitions**.
* Choose **Create new task definition**, **Create new task definition with JSON**.
* In the JSON editor box, copy and paste the following JSON text.

**{**

**"containerDefinitions": [**

**{**

**"memory": 128,**

**"portMappings": [**

**{**

**"hostPort": 80,**

**"containerPort": 80,**

**"protocol": "tcp"**

**}**

**],**

**"essential": true,**

**"mountPoints": [**

**{**

**"containerPath": "/usr/share/nginx/html",**

**"sourceVolume": "efs-html"**

**}**

**],**

**"name": "nginx",**

**"image": "nginx"**

**}**

**],**

**"volumes": [**

**{**

**"name": "efs-html",**

**"efsVolumeConfiguration": {**

**"fileSystemId": "fs-1324abcd",**

**"transitEncryption": "ENABLED"**

**}**

**}**

**],**

**"family": "efs-tutorial",**

**"executionRoleArn":"arn:aws::iam::111122223333:role/ecsTaskExecutionRole"**

**}**

Replace the “filesystemID” with Central-EFS ID (our EFS created in previous step).

* Choose **Create**.

**Explanation of each part of JSON:**

https://chatgpt.com/share/66f00e81-a3c8-8001-b234-c352e51dac6c

**7. Run a task and view the results**

Now that your Amazon EFS file system is created and there is web content for the NGINX container to serve, you can run a task using the task definition that you created. The NGINX web server serves your simple HTML page. If you update the content in your Amazon EFS file system, those changes are propagated to any containers that have also mounted that file system.

* Open the ECS console.
* On the **Clusters** page, select the cluster to create the service in.
* From the **Tasks** tab, choose **Run new task**.
* In the **Compute options** section, select **Launch type**.
* For **Launch type**, choose **EC2**.
* For **Application type**, choose **Task**.
* For **Task definition**, choose the efs-tutorial task definition that you created earlier.
* For **Desired tasks**, enter 1.
* Choose **Create**.
* On the **Cluster** page, choose **Infrastructure**.
* Under **Container Instances**, choose the container instance to connect to.
* On the **Container Instance** page, under **Networking**, record the **Public IP** for your instance.
* Open a browser and enter the public IP address. You should see the following message:

