

Final Project Group 2

Project Overview: Running Web application (Twitter application) in ECS and performing schedule based auto scaling of tasks and ECS instances.

Cloud provider : AWS

Aws Components:

1. Code Build
2. Load Balancer
3. Target Groups
4. Amazon Elastic Container Service(ECS)

Deployment and application workflow

Code build and Github

1. Push twitter backend and frontend code along with buildspec and docker files to a repository. (Reference link : <https://github.com/dinorows/twtr-jwt>)
2. Create a code build in AWS and use GitHub for the Source provider. Select Connect using OAuth, and click Connect to Github and allow access to your GitHub repo for twtr-jwt.
3. After authenticating, under Repository, select Repository in my GitHub account. Then, add the GitHub repository you created for this project.
4. Environment settings
 - Environment image - use the Managed image
 - Operating system - Ubuntu
 - Runtime - Standard
 - Image - aws/codebuild/standard:4.0
 - Image version - Always use the latest image for this runtime version
 - Privileged - check the flag
 - Service role - New service role
 - Role name - flask-react-build-role
5. Add your AWS account ID and AWS region as environment variables called AWS_ACCOUNT_ID and AWS_REGION in plaintext
6. Under Build specifications, select Use a buildspec file.

Screenshot of Code Build in AWS

The screenshot shows the AWS CodeBuild console. On the left, there's a navigation sidebar with 'Developer Tools' and 'CodeBuild' selected. The main area displays a 'Build projects' list with one item: 'twtr-build'. The details for this project are shown in a table:

Name	Source provider	Repository	Latest build status	Description	Last Modified
twtr-build	GitHub	MaddiSanthosh/twtr-jwt	Succeeded	build and test twtr docker images	1 month ago

Note: If code build fails because of docker throttling add docker login command to the pre build section in build spec file.

Docker images in ECR (Built and pushed by code build)

The screenshot shows the AWS Amazon Elastic Container Registry (ECR) console. On the left, there's a navigation sidebar with 'Amazon Elastic Container Registry' selected. The main area displays a 'Private repositories' list with two items: 'twtr-be' and 'twtr-fe'. The details for these repositories are shown in a table:

Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type	Pull through cache
twtr-be	561474959970.dkr.ecr.us-east-1.amazonaws.com/twtr-be	October 31, 2022, 17:06:44 (UTC-04)	Disabled	Manual	AES-256	Inactive
twtr-fe	561474959970.dkr.ecr.us-east-1.amazonaws.com/twtr-fe	October 31, 2022, 17:06:54 (UTC-04)	Disabled	Manual	AES-256	Inactive

Application Load Balancer

1. Configure Load Balancer – Name:flask-react-alb
 - Scheme:internet-facing
 - IP address type: ipv4
 - Listeners: HTTP / Port 80
 - VPC: Select the default VPC to keep things simple
 - Availability Zones: Select at least two available subnets
2. Skip configure security settings
3. Configure security groups : select default security group.
4. Create a target group called “flask-react-fe-tg”, type : instance, Port: 80, Path: “/”
5. Create a target group called “flask-react-be-tg”, type : instance, Port: 5000, Path: “/doc”
6. Register both target groups with load balancer
7. Edit listener rules of the load balancer as shown in below screenshot

The screenshot shows the AWS Application Load Balancer (ALB) Listener Rules configuration page. The top navigation bar includes 'Rules' (selected), a plus sign for creating new rules, edit and delete icons, and tabs for 'flask-react-alb | HTTP:80' and 'HTTPS:443'. A note says 'To edit, select a rule above.' Below this, a message indicates 'flask-react-alb | HTTP:80 (2 rules)'.

A tooltip for 'Rule limits for condition values, wildcards, and total rules.' is visible. The main table displays two rules:

ID	Condition	Action
1 arn...a02f6	IF ✓ Path is /doc OR /tweet* OR /login OR /fastlogin	THEN Forward to flask-react-be-tg: 1 (100%) Group-level stickiness: Off
last HTTP 80: default action <small>This rule cannot be moved or deleted</small>	IF ✓ Requests otherwise not routed	THEN Forward to flask-react-fe-tg: 1 (100%) Group-level stickiness: Off

Target Groups :

The screenshot shows the AWS EC2 Target Groups page. The top navigation bar includes 'EC2 > Target groups' and a 'Create target group' button. A search bar and pagination controls are also present.

The table lists two target groups:

Name	ARN	Port	Protocol	Target type	Load balancer
flask-react-be-tg	arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/flask-react-be-tg/123456789012345678	5000	HTTP	Instance	flask-react-all
flask-react-fe-tg	arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/flask-react-fe-tg/123456789012345678	80	HTTP	Instance	flask-react-all

Load Balancer

The screenshot shows the AWS EC2 Load Balancers details page for a load balancer named "flask-react-alb". The left sidebar includes links for New EC2 Experience, EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (with sub-links for Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Scheduled Instances, Capacity Reservations), Images (AMIs, AMI Catalog), and Elastic Block Store.

Details:

Load balancer type	DNS name	Status	VPC
Application	flask-react-alb-1770583879.us-east-1.elb.amazonaws.com (A Record)	Active	vpc-037cb5ece6a082cee
IP address type	Scheme	Availability Zones	Hosted Zone
IPv4	Internet-facing	subnet-03ceb1797bd96a7a0 us-east-1a (use1-az6) subnet-0bb30b59aa1b3ff9e us-east-1b (use1-az1)	Z35SXDOTRQ7X7K
Created At			
December 9, 2022, 18:10 (UTC-05:00)			

Listeners: (1) [Edit](#) [Actions ▾](#) [Add listener](#)

Note : Get the DNS of the load balancer from Load balancer details page. Replace the url (REACT_APP_API_SERVICE_URL) in build spec file with load balancer DNS. Make sure that code build triggers after updating build spec file.

AWS ECS

Create Ecs cluster

1. Select EC2 Linux + Networking
2. Cluster name: flask-react-cluster
3. EC2 instance type: t2.micro
4. Number of instances: 2
5. Select default VPC(same as load balancer vpc)
6. Select existing key pair or create new key pair

Cluster

The screenshot shows the AWS ECS Cluster details page for a cluster named 'twtr'. The top navigation bar includes links for AWS, Services, Search, and regions N. Virginia and maddisanthosh. The left sidebar lists various AWS services: Task Definitions, Account Settings, Amazon EKS, Clusters, Amazon ECR, Repositories, AWS Marketplace, Discover software, and Subscriptions. The main content area displays the cluster's ARN (arn:aws:ecs:us-east-1:561474959970:cluster/twtr), status (ACTIVE), and registered container instances (5). It also shows task counts: Pending tasks count (0 Fargate, 0 EC2, 0 External), Running tasks count (0 Fargate, 5 EC2, 0 External), Active service count (0 Fargate, 2 EC2, 0 External), and Draining service count (0 Fargate, 0 EC2, 0 External). Below this, a table lists two services: 'fe' and 'be'. The 'fe' service is active, replicating tasks on EC2 with a desired count of 1 and running count of 1. The 'be' service is active, replicating tasks on EC2 with a desired count of 5 and running count of 4. The table includes columns for Service Name, Status, Service type, Task Definition, Desired tasks, Running tasks, Launch type, and Platform version.

Service Name	Status	Service type	Task Definition	Desired tasks	Running tasks	Launch type	Platform version
fe	ACTIVE	REPLICA	flask-react-f...	1	1	EC2	--
be	ACTIVE	REPLICA	flask-react-...	5	4	EC2	--

Create frontend Task Definitions:

1. Select EC2 in the Select launch type compatibility screen
2. Set Task Definition Name to flask-react-fe-td Task Role: do not select
3. Network Mode: do not select
4. Task execution IAM role: ecsTaskExecutionRole
5. Add container: Container name: fe
6. Image: <YOUR_AWS_ACCOUNT_ID>.dkr.ecr.us-east-1.amazonaws.com/twtr-fe:prod
7. Memory Limits (MB): 500 soft limit
8. Port mappings: 0 or empty host, 80 container

Create backend Task Definitions:

1. select EC2 in the Select launch type compatibility screen
2. Set Task Definition Name to flask-react-be-td Task Role: do not select
3. Network Mode: do not select
4. Task execution IAM role: ecsTaskExecutionRole
5. Add container: Container name: be
6. Image: <YOUR_AWS_ACCOUNT_ID>.dkr.ecr.us-east-1.amazonaws.com/twtr-be:prod
7. Memory Limits (MB): 500 soft limit
8. Port mappings: 0 or empty host, 5000 container

Task Definitions:

The screenshot shows the AWS ECS Task Definitions page. The left sidebar has a 'Task Definitions' section selected. The main content area displays two task definitions:

Task Definition	Latest revision status
flask-react-be-td	ACTIVE
flask-react-fe-td	ACTIVE

Create backend Service:

1. Launch type: EC2
2. Task Definition:
 - a. – Family:flask-react-be-td
 - b. – Revision:LATEST_REVISION_NUMBER
3. Service name: flask-react-be-service Number of tasks: 1
4. Select the Application Load Balancer under Load balancer type
5. Load balancer name: flask-react-alb Container name: port: be:0:5000
6. Click Add to load balancer
7. Production listener port: 80:HTTP
8. Target group name: flask-react-be-tg

Create frontend Service:

1. Launch type: EC2
2. Task Definition:
 - a. – Family:flask-react-fe-td
 - b. – Revision:LATEST_REVISION_NUMBER
3. Service name: flask-react-fe-service, Number of tasks: 1
4. Select the Application Load Balancer under Load balancer type
5. Load balancer name: flask-react-alb Container name: port: be:0:80
6. Click Add to load balancer
7. Production listener port: 80:HTTP
8. Target group name: flask-react-fe-tg

Services

Cluster : twtr

Service Name	Status	Service ty...	Task Defini...	Desired ta...	Running ta...	Launch ty...	Platform v...
fe	ACTIVE	REPLIC...	flask-react-f...	1	1	EC2	--
be	ACTIVE	REPLIC...	flask-react-...	5	4	EC2	--

Note: Verify that new tasks are created and status is running as shown in below image.

TASKS:

The screenshot shows the AWS CloudWatch Tasks page for a cluster named 'twtr'. The cluster ARN is listed as arn:aws:ecs:us-east-1:561474959970:cluster/twtr. The status is ACTIVE. There are 5 registered container instances. Task statistics show 0 Pending tasks, 0 Running tasks, 0 Active service count, and 0 Draining service count. Below the statistics, there are tabs for Services, Tasks (which is selected), ECS Instances, Metrics, Scheduled Tasks, Tags, and Capacity Providers. A 'Run new Task' button is available. The main table lists 5 tasks, all of which are currently RUNNING. The columns include Task ID, Task definition, Container ID, Last status, Desired state, Started at, Started By, Group, Launch type, and Platform version. The tasks are: 480ffdfe13..., 6e66005117..., 7c81e14829..., 8a0ab28ed5..., and d2350533f6... All tasks are associated with the 'service:be' group and run on EC2 platforms.

Note: In Ec2's security groups page select the Security Group associated with the containers, and allow traffic with selecting source as ALB's security group.

FRONTEND:(Access using DNS of load balancer)

ATLAS:

The screenshot shows the MongoDB Atlas Data Services interface. On the left, a sidebar includes sections for Deployment, Database (with a Data Lake PREVIEW), Data Services (Triggers, Data API, Data Federation), and Security (Database Access, Network Access, Advanced). The main area is titled 'tweets.tweets' and shows a preview of the collection. It displays storage details: STORAGE SIZE: 36KB, LOGICAL DATA SIZE: 2.78KB, TOTAL DOCUMENTS: 16, and INDEXES TOTAL SIZE: 36KB. Below this, there are tabs for Find, Indexes, Schema Anti-Patterns, Aggregation, and Search Indexes. A 'FILTER' bar is present with the query '{ field: 'value' }'. The results table shows 16 documents, with the first few entries being:

```

1 _id: "6397a3b9b7943d151d1dc477"
2 user: "group_2"
3 description: "Final demo"
4 private: true
5 upvote: 0
6 date: "2022-12-12 21:15:51"
7 pic: "https://randomuser.me/api/portraits/women/45.jpg"

```

Each document entry includes a row number, the '_id' field, and the 'user', 'description', 'private', 'upvote', 'date', and 'pic' fields. To the right of the table, column types are listed: String, String, String, Boolean, Int32, and String. At the bottom of the interface, it says 'System Status: All Good'.

AutoScaling

CAPACITY PROVIDERS:

1. Create a capacity provider in the twitter cluster and select the auto scaling group associated with cluster.

The screenshot shows the AWS CloudWatch Metrics console with the 'Capacity Providers' tab selected for the 'twtr' cluster. The left sidebar includes links for Amazon ECR, Repositories, AWS Marketplace, Discover software, and Subscriptions. The main area displays cluster statistics and a table of capacity providers.

Capacity Pr...	Type	ASG	Managed S...	Managed I...	Current Siz...	Desired Siz...	Min Size	Max Size	Update Sta...
cp-flask	ASGProvider	EC2Contain...	Yes	No	5	5	5	5	

2. Go the auto scaling group and create a scheduled action and set desire capacity to 5 and time stamp to the current time stamp and submit
3. Go back to the cluster and verify that cluster has 5 ECS instances.

Verify the scheduled action status in activity of ASG as shown below

Capacity Reservations		Activity			Last updated on December 12, 2022 6:20:05 PM (0m ago)	
▼ Images		InProgress	instance: i-046db33a88a452a2f	update of AutoScalingGroup constraints to min: 3, max: 3, desired: 3 changing the desired capacity from 4 to 3. At 2022-12-12T21:19:15Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 4 to 3. At 2022-12-12T21:19:15Z instance i-046db33a88a452a2f was selected for termination.	Decem	12, 04 PM -0!
AMIs		Successful	Executing scheduled action scalein		2022	Decem
AMI Catalog		Successful	Launching a new EC2 instance: i-02a0972c775d31c02	At 2022-12-12T21:16:00Z the scheduled action changingto4 executed. Setting min size from 3 to 4. Setting max size from 3 to 4. Setting desired capacity from 3 to 4. At 2022-12-12T21:16:00Z a scheduled action update of AutoScalingGroup constraints to min: 4, max: 4, desired: 4 changing the desired capacity from 3 to 4. At 2022-12-12T21:16:09Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 3 to 4.	Decem	12, 04 PM -0!
▼ Elastic Block Store		Successful	Executing scheduled action changingto4		2022	Decem
Volumes		Successful	Launching a new EC2 instance: i-046db33a88a452a2f	At 2022-12-12T16:38:03Z a scheduled action update of AutoScalingGroup constraints to min: 3, max: 3, desired: 3 changing the desired capacity from 2 to 3. At 2022-12-12T16:38:03Z the scheduled action test executed. Setting min size from 0 to 3. Setting max size from 2 to 3. Setting desired capacity from 2 to 3. At 2022-12-12T16:38:14Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 2 to 3.	Decem	12, 11 AM -0!
Snapshots						
Lifecycle Manager						
▼ Network & Security						
Security Groups						
Elastic IPs						
Placement Groups						
Key Pairs						
Network Interfaces						
▼ Load Balancing						
Load Balancers New						
Target Groups New						

ECS INSTANCES:

The screenshot shows the AWS ECS Instances page. At the top, it displays "Draining service count 0 Fargate, 0 EC2, 0 External". Below this is a navigation bar with tabs: Services, Tasks, **ECS Instances**, Metrics, Scheduled Tasks, Tags, and Capacity Providers. A message box provides instructions for registering External and Amazon EC2 instances. The main content area includes a "Register External Instances" button, an "Actions" dropdown, and a table of registered instances. The table has columns: Container Instance, ECS Instance ID, Availability Zone, External Instance ID, Agent Connected, Status, Running tasks, and CPU available. The table lists six instances, all in the ACTIVE state.

	Container Instance	ECS Instance ...	Availability Zo...	External Insta...	Agent Connec...	Status	Running tasks...	CPU available
<input type="checkbox"/>	19d6e185bfe40b491abd3...	i-0fd5b78662c7...	us-east-1b	false	true	ACTIVE	1	1024
<input type="checkbox"/>	5a45c971259a4ef4a461af...	i-015d1f1ff8652...	us-east-1b	false	true	ACTIVE	1	1024
<input type="checkbox"/>	5f17065e31284334869256...	i-026183de2c3...	us-east-1a	false	true	ACTIVE	1	1024
<input type="checkbox"/>	694f937623354d21aac770...	i-0e8e4867af4...	us-east-1a	false	true	ACTIVE	1	1024
<input type="checkbox"/>	6b12517601cc4b9480291...	i-02a0972c775...	us-east-1a	false	true	ACTIVE	1	1024

Auto scaling Service Tasks:

Use below AWS CLI commands to scale n and scale out tasks of a service .

1. Register target :

```
aws application-autoscaling register-scalable-target --service-namespace ecs  
--scalable-dimension ecs:service:DesiredCount --resource-id  
service/<cluset-name>/<service-name> --min-capacity 1 --max-capacity 10 --region  
us-east-1
```

2. Put scheduled action

```
aws application-autoscaling put-scheduled-action --service-namespace ecs  
--scalable-dimension ecs:service:DesiredCount --resource-id  
service/<cluset-name>/<service-name> --scheduled-action-name single-scaleout-action  
--schedule "at(2022-12-12T21:17:00)" --scalable-target-action  
MinCapacity=1,MaxCapacity=1 --region us-east-1
```

After running the command by providing correct cluster name, service name and current time stamp go back to the service and verify that number of task are changed as expected.

References:

1. <https://aws.amazon.com/blogs/containers/optimizing-amazon-elastic-container-service-for-cost-using-scheduled-scaling/>
- 2.