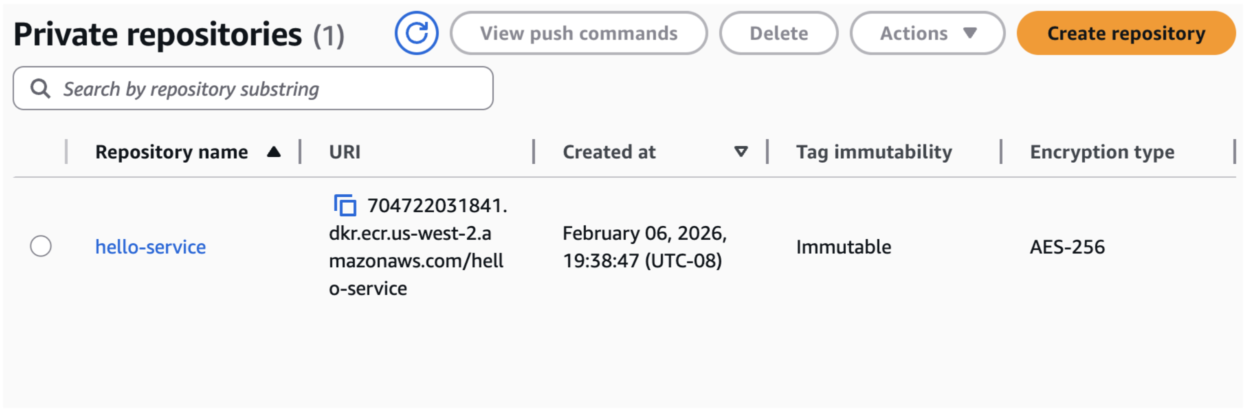
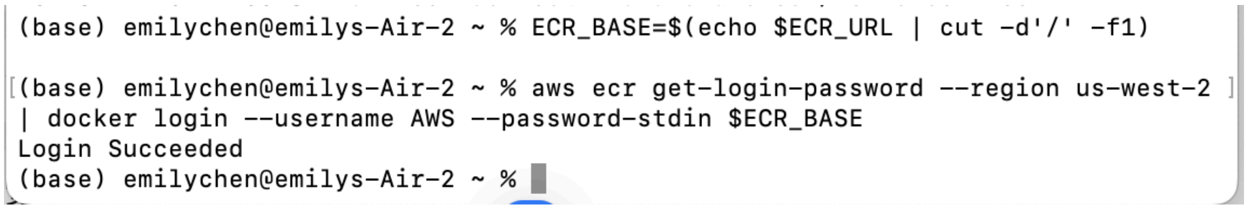
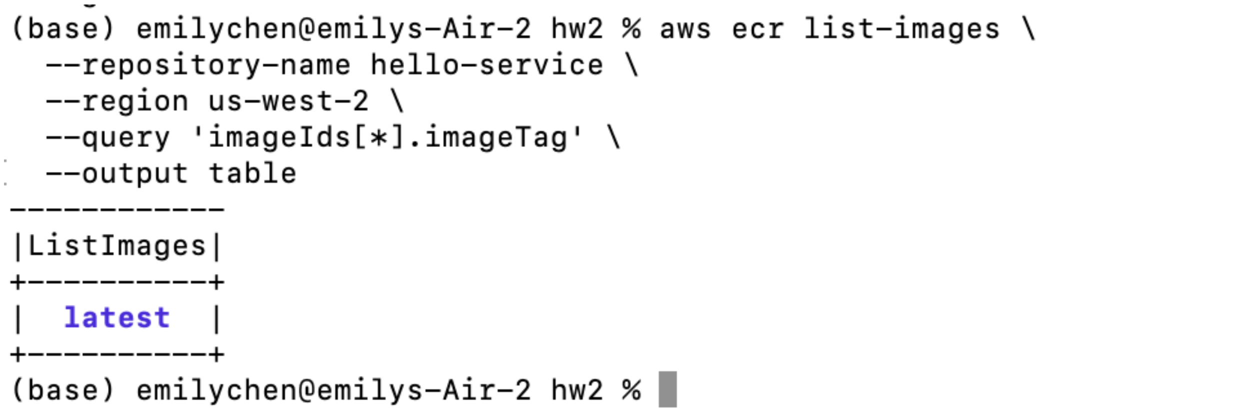
Part II



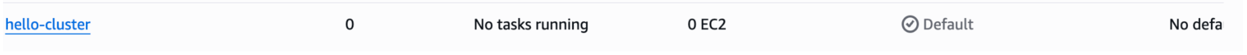
Repository name	URI	Created at	Tag immutability	Encryption type
hello-service	704722031841.dkr.ecr.us-west-2.amazonaws.com/hello-service	February 06, 2026, 19:38:47 (UTC-08)	Immutable	AES-256



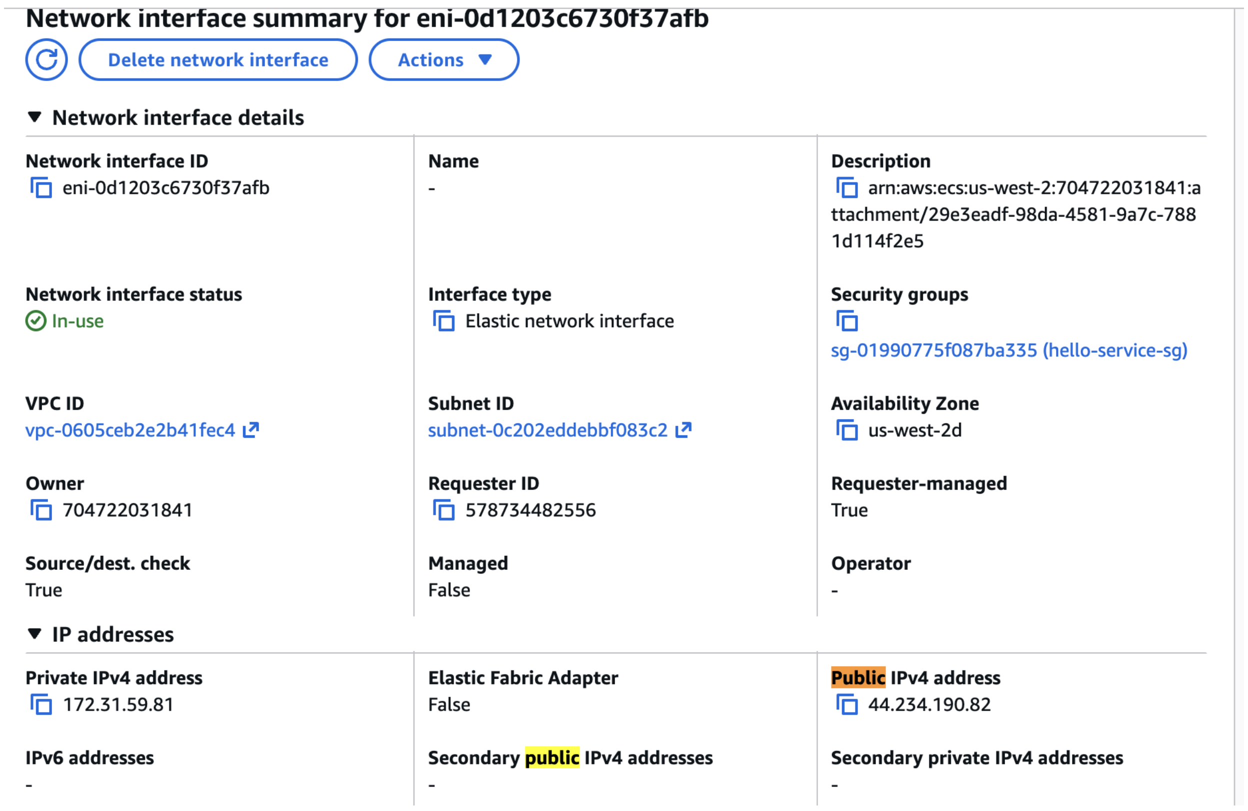
```
(base) emilychen@emilys-Air-2 ~ % ECR_BASE=$(echo $ECR_URL | cut -d'/' -f1)

(base) emilychen@emilys-Air-2 ~ % aws ecr get-login-password --region us-west-2
| docker login --username AWS --password-stdin $ECR_BASE
Login Succeeded
(base) emilychen@emilys-Air-2 ~ %
```



```
(base) emilychen@emilys-Air-2 hw2 % aws ecr list-images \
  --repository-name hello-service \
  --region us-west-2 \
  --query 'imageIds[*].imageTag' \
  --output table
------------
|ListImages|
+----------+
|  latest  |
+----------+
(base) emilychen@emilys-Air-2 hw2 %
```



hello-cluster	0	No tasks running	0 EC2	✓ Default	No defa



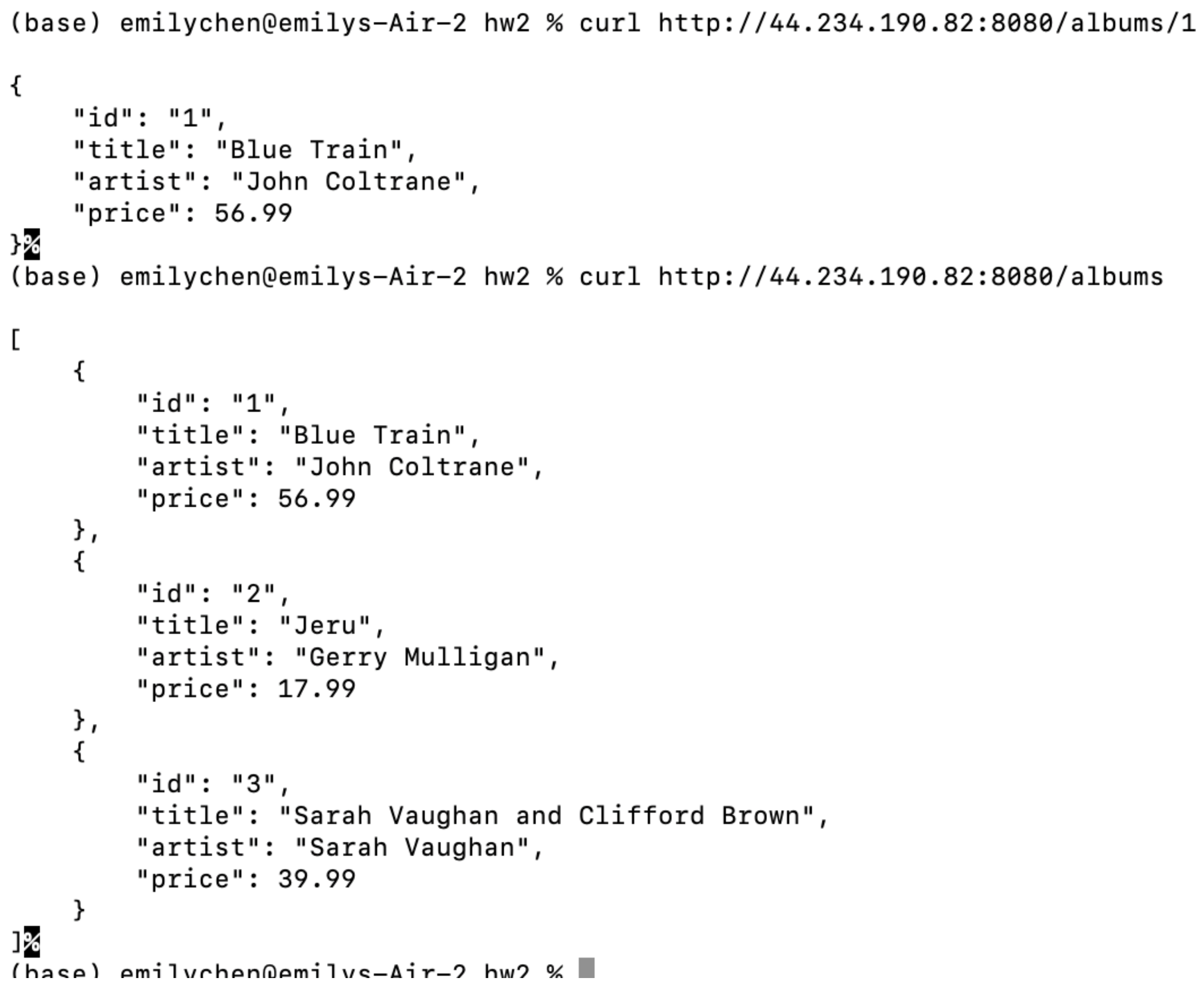
Network interface summary for eni-0d1203c6730f37afb

Network interface details

Network interface ID eni-0d1203c6730f37afb	**Name** -	**Description** arn:aws:ecs:us-west-2:704722031841:attachment/29e3eadf-98da-4581-9a7c-7881d114f2e5
Network interface status In-use	**Interface type** Elastic network interface	**Security groups** sg-01990775f087ba335 (hello-service-sg)
VPC ID vpc-0605ceb2e2b41fec4	**Subnet ID** subnet-0c202eddebbf083c2	**Availability Zone** us-west-2d
Owner 704722031841	**Requester ID** 578734482556	**Requester-managed** True
Source/dest. check True	**Managed** False	**Operator** -

IP addresses

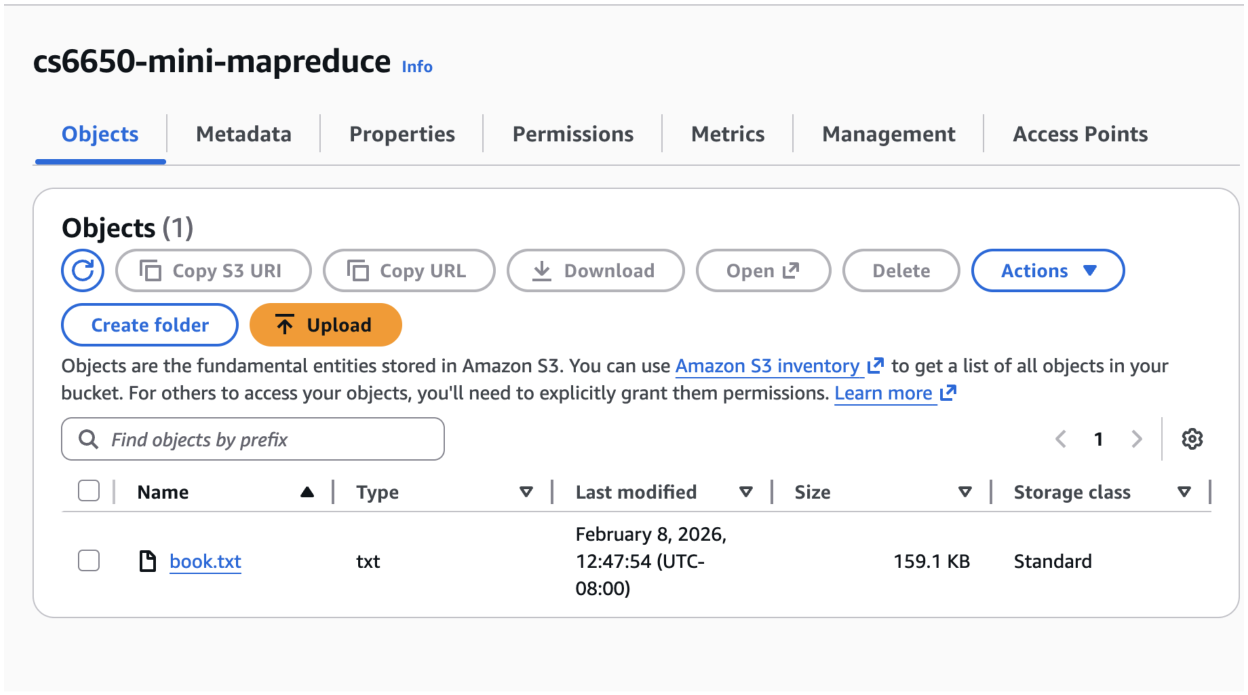
Private IPv4 address 172.31.59.81	**Elastic Fabric Adapter** False	**Public IPv4 address** 44.234.190.82
IPv6 addresses -	**Secondary public IPv4 addresses** -	**Secondary private IPv4 addresses** -



```
(base) emilychen@emilys-Air-2 hw2 % curl http://44.234.190.82:8080/albums/1

{
    "id": "1",
    "title": "Blue Train",
    "artist": "John Coltrane",
    "price": 56.99
}%
(base) emilychen@emilys-Air-2 hw2 % curl http://44.234.190.82:8080/albums

[
    {
        "id": "1",
        "title": "Blue Train",
        "artist": "John Coltrane",
        "price": 56.99
    },
    {
        "id": "2",
        "title": "Jeru",
        "artist": "Gerry Mulligan",
        "price": 17.99
    },
    {
        "id": "3",
        "title": "Sarah Vaughan and Clifford Brown",
        "artist": "Sarah Vaughan",
        "price": 39.99
    }
]%
(base) emilychen@emilys-Air-2 hw2 %
```

Part III - MapReducer

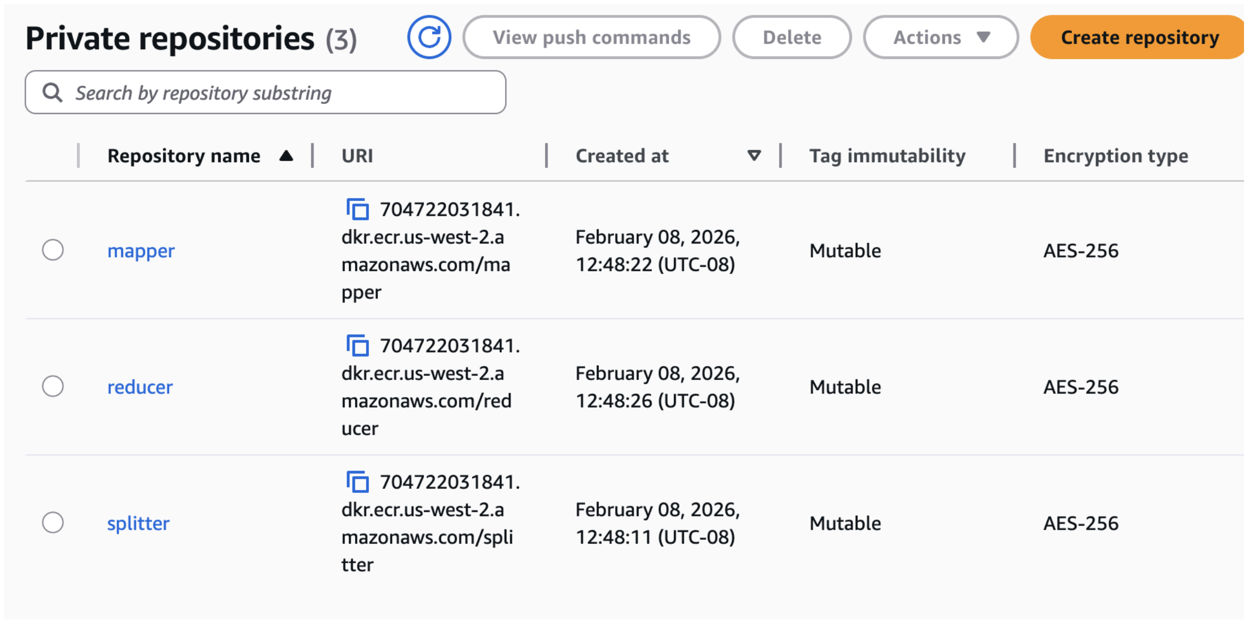
1. Create a S3 Bucket and upload the file



Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
book.txt	txt	February 8, 2026, 12:47:54 (UTC-08:00)	159.1 KB	Standard

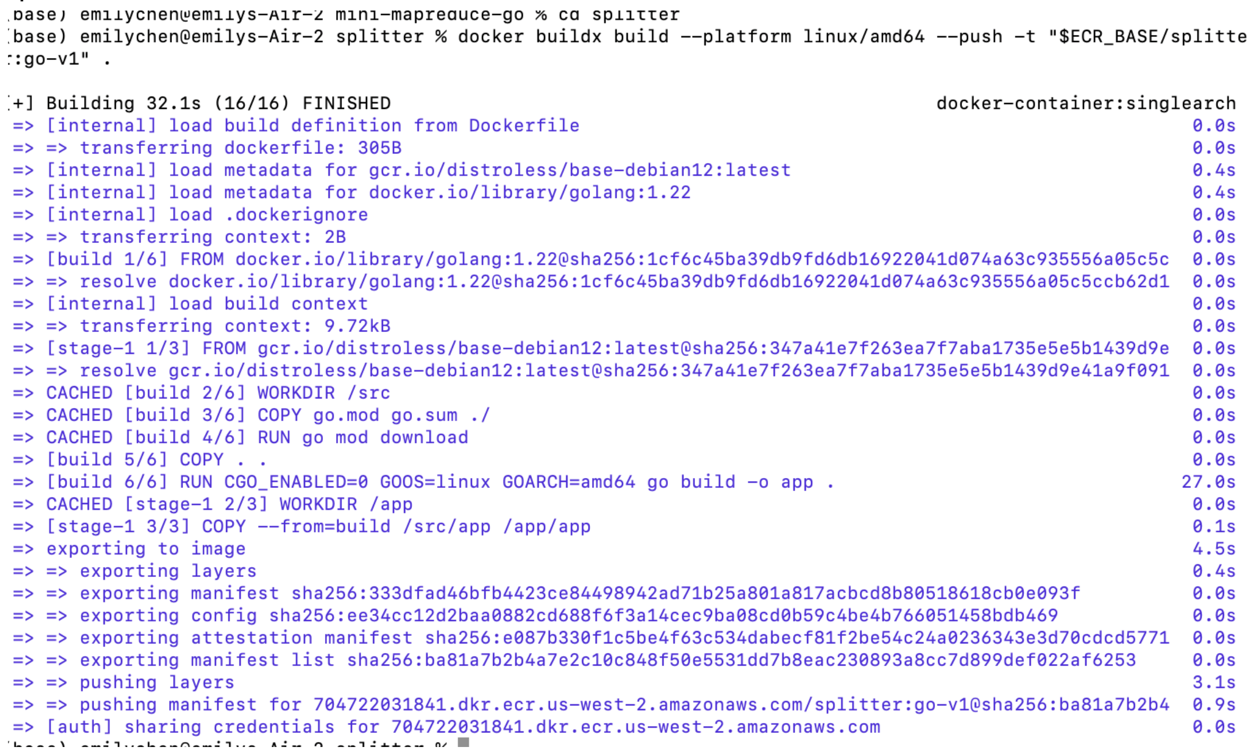
2. Create 3 ECR Repos



Repository name	URI	Created at	Tag immutability	Encryption type
mapper	704722031841.dkr.ecr.us-west-2.amazonaws.com/mapper	February 08, 2026, 12:48:22 (UTC-08)	Mutable	AES-256
reducer	704722031841.dkr.ecr.us-west-2.amazonaws.com/reducer	February 08, 2026, 12:48:26 (UTC-08)	Mutable	AES-256
splitter	704722031841.dkr.ecr.us-west-2.amazonaws.com/splitter	February 08, 2026, 12:48:11 (UTC-08)	Mutable	AES-256

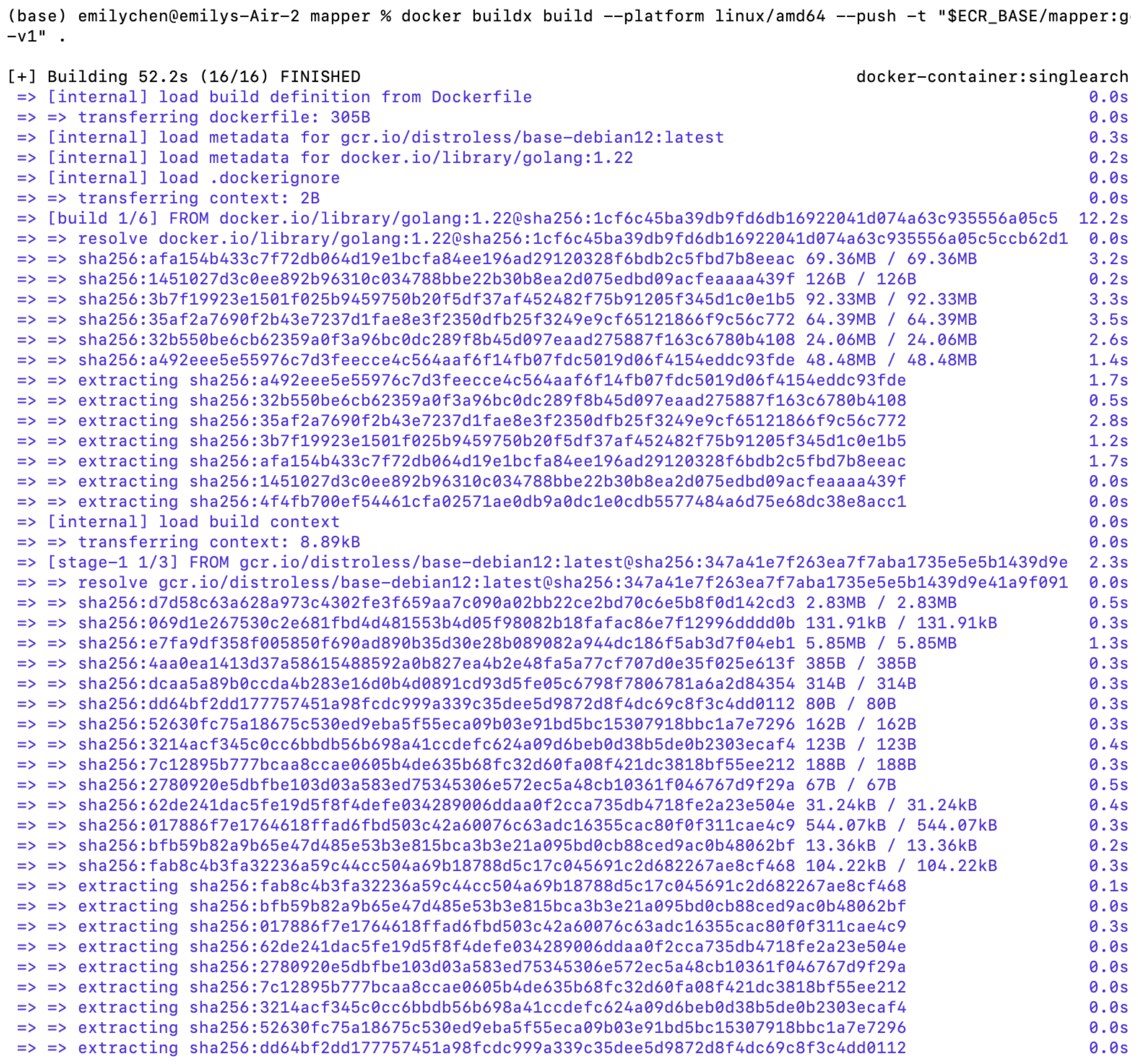
3. Push docker image to ECR

splitter:



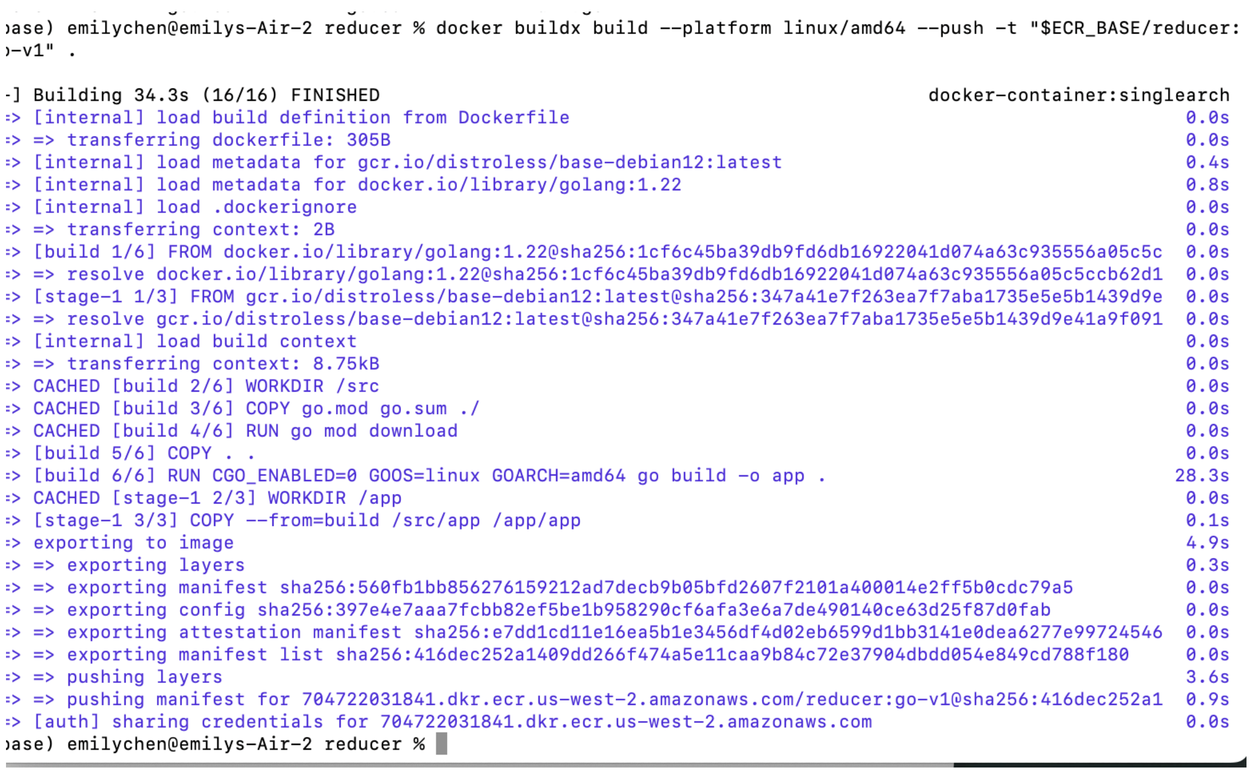
```
base) emilychen@emilys-Air-2 mini-mapreduce-go % cd splitter
base) emilychen@emilys-Air-2 splitter % docker buildx build --platform linux/amd64 --push -t "$ECR_BASE/splitte
:go-v1" .

+] Building 32.1s (16/16) FINISHED                                                   docker-container:singlearch
 => [internal] load build definition from Dockerfile                                                        0.0s
 => => transferring dockerfile: 305B                                                                        0.0s
 => [internal] load metadata for gcr.io/distroless/base-debian12:latest                                     0.4s
 => [internal] load metadata for docker.io/library/golang:1.22                                              0.4s
 => [internal] load .dockerignore                                                                           0.0s
 => => transferring context: 2B                                                                             0.0s
 => [build 1/6] FROM docker.io/library/golang:1.22@sha256:1cf6c45ba39db9fd6db16922041d074a63c935556a05c5c  0.0s
 => => resolve docker.io/library/golang:1.22@sha256:1cf6c45ba39db9fd6db16922041d074a63c935556a05c5ccb62d1  0.0s
 => [internal] load build context                                                                           0.0s
 => => transferring context: 9.72kB                                                                         0.0s
 => [stage-1 1/3] FROM gcr.io/distroless/base-debian12:latest@sha256:347a41e7f263ea7f7aba1735e5e5b1439d9e  0.0s
 => => resolve gcr.io/distroless/base-debian12:latest@sha256:347a41e7f263ea7f7aba1735e5e5b1439d9e41a9f091  0.0s
 => CACHED [build 2/6] WORKDIR /src                                                                         0.0s
 => CACHED [build 3/6] COPY go.mod go.sum ./                                                                0.0s
 => CACHED [build 4/6] RUN go mod download                                                                  0.0s
 => [build 5/6] COPY . .                                                                                    0.0s
 => [build 6/6] RUN CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build -o app .                                27.0s
 => CACHED [stage-1 2/3] WORKDIR /app                                                                       0.0s
 => [stage-1 3/3] COPY --from=build /src/app /app/app                                                       0.1s
 => exporting to image                                                                                      4.5s
 => => exporting layers                                                                                     0.4s
 => => exporting manifest sha256:333dfad46bfb4423ce84498942ad71b25a801a817acbcd8b80518618cb0e093f           0.0s
 => => exporting config sha256:ee34cc12d2baa0882cd688f6f3a14cec9ba08cd0b59c4be4b766051458bdb469             0.0s
 => => exporting attestation manifest sha256:e087b330f1c5be4f63c534dabecf81f2be54c24a0236343e3d70cdcd5771  0.0s
 => => exporting manifest list sha256:ba81a7b2b4a7e2c10c848f50e5531dd7b8eac230893a8cc7d899def022af6253      0.0s
 => => pushing layers                                                                                       3.1s
 => => pushing manifest for 704722031841.dkr.ecr.us-west-2.amazonaws.com/splitter:go-v1@sha256:ba81a7b2b4  0.9s
 => [auth] sharing credentials for 704722031841.dkr.ecr.us-west-2.amazonaws.com                             0.0s
base) emilychen@emilys-Air-2 splitter %
```

Mapper:

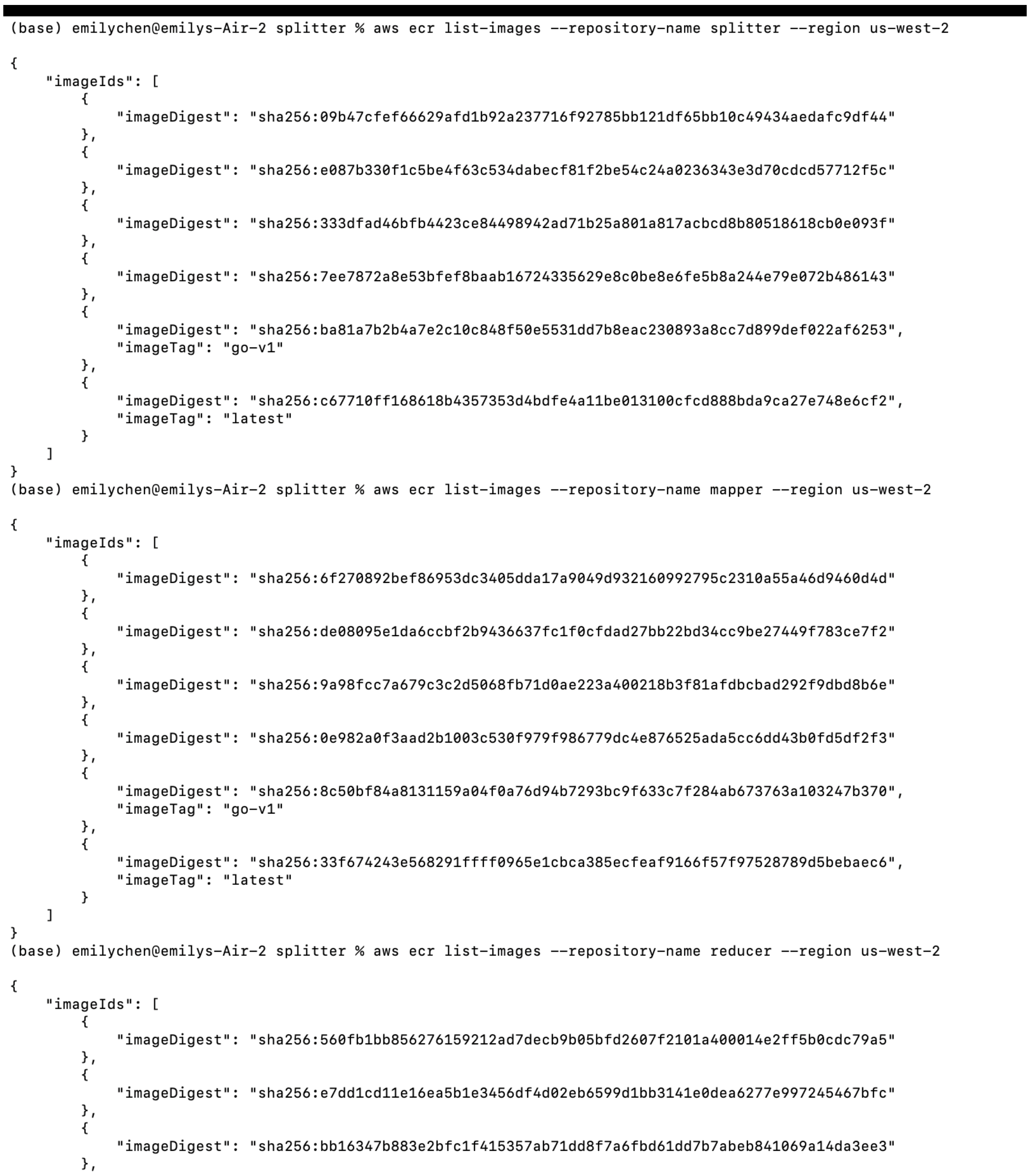


```
(base) emilychen@emilys-Air-2 mapper % docker buildx build --platform linux/amd64 --push -t "$ECR_BASE/mapper:g
-v1" .

[+] Building 52.2s (16/16) FINISHED                                                    docker-container:singlearch
 => [internal] load build definition from Dockerfile                                                         0.0s
 => => transferring dockerfile: 305B                                                                         0.0s
 => [internal] load metadata for gcr.io/distroless/base-debian12:latest                                      0.3s
 => [internal] load metadata for docker.io/library/golang:1.22                                               0.2s
 => [internal] load .dockerignore                                                                            0.0s
 => => transferring context: 2B                                                                              0.0s
 => [build 1/6] FROM docker.io/library/golang:1.22@sha256:1cf6c45ba39db9fd6db16922041d074a63c935556a05c5  12.2s
 => => resolve docker.io/library/golang:1.22@sha256:1cf6c45ba39db9fd6db16922041d074a63c935556a05c5ccb62d1  0.0s
 => => sha256:afa154b433c7f72db064d19e1bcfa84ee196ad29120328f6bdb2c5fbd7b8eeac 69.36MB / 69.36MB            3.2s
 => => sha256:1451027d3c0ee892b96310c034788bbe22b30b8ea2d075edbd09acfeaaaa439f 126B / 126B                  0.2s
 => => sha256:3b7f19923e1501f025b9459750b20f5df37af452482f75b91205f345d1c0e1b5 92.33MB / 92.33MB            3.3s
 => => sha256:35af2a7690f2b43e7237d1fae8e3f2350dfb25f3249e9cf65121866f9c56c772 64.39MB / 64.39MB            3.5s
 => => sha256:32b550be6cb62359a0f3a96bc0dc289f8b45d097eaad275887f163c6780b4108 24.06MB / 24.06MB            2.6s
 => => sha256:a492eee5e55976c7d3feecce4c564aaf6f14fb07fdc5019d06f4154eddc93fde 48.48MB / 48.48MB            1.4s
 => => extracting sha256:a492eee5e55976c7d3feecce4c564aaf6f14fb07fdc5019d06f4154eddc93fde                   1.7s
 => => extracting sha256:32b550be6cb62359a0f3a96bc0dc289f8b45d097eaad275887f163c6780b4108                   0.5s
 => => extracting sha256:35af2a7690f2b43e7237d1fae8e3f2350dfb25f3249e9cf65121866f9c56c772                   2.8s
 => => extracting sha256:3b7f19923e1501f025b9459750b20f5df37af452482f75b91205f345d1c0e1b5                   1.2s
 => => extracting sha256:afa154b433c7f72db064d19e1bcfa84ee196ad29120328f6bdb2c5fbd7b8eeac                   1.7s
 => => extracting sha256:1451027d3c0ee892b96310c034788bbe22b30b8ea2d075edbd09acfeaaaa439f                   0.0s
 => => extracting sha256:4f4fb700ef54461cfa02571ae0db9a0dc1e0cdb5577484a6d75e68dc38e8acc1                   0.0s
 => [internal] load build context                                                                            0.0s
 => => transferring context: 8.89kB                                                                          0.0s
 => [stage-1 1/3] FROM gcr.io/distroless/base-debian12:latest@sha256:347a41e7f263ea7f7aba1735e5e5b1439d9e   2.3s
 => => resolve gcr.io/distroless/base-debian12:latest@sha256:347a41e7f263ea7f7aba1735e5e5b1439d9e41a9f091  0.0s
 => => sha256:d7d58c63a628a973c4302fe3f659aa7c090a02bb22ce2bd70c6e5b8f0d142cd3 2.83MB / 2.83MB              0.5s
 => => sha256:069d1e267530c2e681fbd4d481553b4d05f98082b18fafac86e7f12996dddd0b 131.91kB / 131.91kB          0.3s
 => => sha256:e7fa9df358f005850f690ad890b35d30e28b089082a944dc186f5ab3d7f04eb1 5.85MB / 5.85MB              1.3s
 => => sha256:4aa0ea1413d37a58615488592a0b827ea4b2e48fa5a77cf707d0e35f025e613f 385B / 385B                  0.3s
 => => sha256:dcaa5a89b0ccda4b283e16d0b4d0891cd93d5fe05c6798f7806781a6a2d84354 314B / 314B                  0.3s
 => => sha256:dd64bf2dd177757451a98fcdc999a339c35dee5d9872d8f4dc69c8f3c4dd0112 80B / 80B                    0.3s
 => => sha256:52630fc75a18675c530ed9eba5f55eca09b03e91bd5bc15307918bbc1a7e7296 162B / 162B                  0.3s
 => => sha256:3214acf345c0cc6bbdb56b698a41ccdefc624a09d6beb0d38b5de0b2303ecaf4 123B / 123B                  0.4s
 => => sha256:7c12895b777bcaa8ccae0605b4de635b68fc32d60fa08f421dc3818bf55ee212 188B / 188B                  0.3s
 => => sha256:2780920e5dbfbe103d03a583ed75345306e572ec5a48cb10361f046767d9f29a 67B / 67B                    0.5s
 => => sha256:62de241dac5fe19d5f8f4defe034289006ddaa0f2cca735db4718fe2a23e504e 31.24kB / 31.24kB            0.4s
 => => sha256:017886f7e1764618ffad6fbd503c42a60076c63adc16355cac80f0f311cae4c9 544.07kB / 544.07kB          0.3s
 => => sha256:bfb59b82a9b65e47d485e53b3e815bca3b3e21a095bd0cb88ced9ac0b48062bf 13.36kB / 13.36kB            0.2s
 => => sha256:fab8c4b3fa32236a59c44cc504a69b18788d5c17c045691c2d682267ae8cf468 104.22kB / 104.22kB          0.3s
 => => extracting sha256:fab8c4b3fa32236a59c44cc504a69b18788d5c17c045691c2d682267ae8cf468                   0.1s
 => => extracting sha256:bfb59b82a9b65e47d485e53b3e815bca3b3e21a095bd0cb88ced9ac0b48062bf                   0.0s
 => => extracting sha256:017886f7e1764618ffad6fbd503c42a60076c63adc16355cac80f0f311cae4c9                   0.3s
 => => extracting sha256:62de241dac5fe19d5f8f4defe034289006ddaa0f2cca735db4718fe2a23e504e                   0.0s
 => => extracting sha256:2780920e5dbfbe103d03a583ed75345306e572ec5a48cb10361f046767d9f29a                   0.0s
 => => extracting sha256:7c12895b777bcaa8ccae0605b4de635b68fc32d60fa08f421dc3818bf55ee212                   0.0s
 => => extracting sha256:3214acf345c0cc6bbdb56b698a41ccdefc624a09d6beb0d38b5de0b2303ecaf4                   0.0s
 => => extracting sha256:52630fc75a18675c530ed9eba5f55eca09b03e91bd5bc15307918bbc1a7e7296                   0.0s
 => => extracting sha256:dd64bf2dd177757451a98fcdc999a339c35dee5d9872d8f4dc69c8f3c4dd0112                   0.0s
```

Reducer:

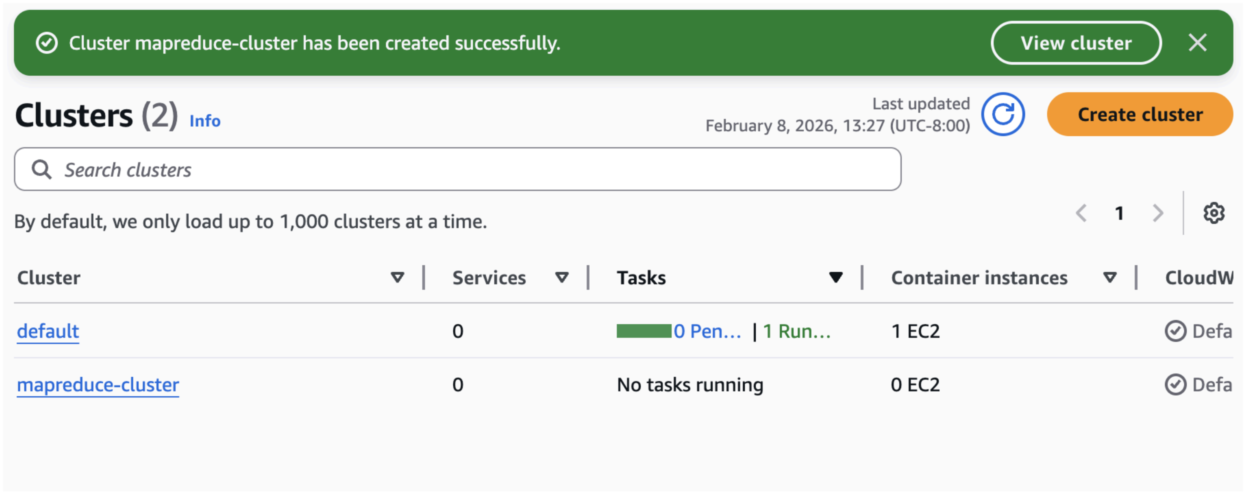


```
ɔase) emilychen@emilys-Air-2 reducer % docker buildx build --platform linux/amd64 --push -t "$ECR_BASE/reducer:
ɔ-v1" .

-] Building 34.3s (16/16) FINISHED                                                    docker-container:singlearch
=> [internal] load build definition from Dockerfile                                                          0.0s
=> => transferring dockerfile: 305B                                                                          0.0s
=> [internal] load metadata for gcr.io/distroless/base-debian12:latest                                       0.4s
=> [internal] load metadata for docker.io/library/golang:1.22                                                0.8s
=> [internal] load .dockerignore                                                                             0.0s
=> => transferring context: 2B                                                                               0.0s
=> [build 1/6] FROM docker.io/library/golang:1.22@sha256:1cf6c45ba39db9fd6db16922041d074a63c935556a05c5c  0.0s
=> => resolve docker.io/library/golang:1.22@sha256:1cf6c45ba39db9fd6db16922041d074a63c935556a05c5ccb62d1  0.0s
=> [stage-1 1/3] FROM gcr.io/distroless/base-debian12:latest@sha256:347a41e7f263ea7f7aba1735e5e5b1439d9e  0.0s
=> => resolve gcr.io/distroless/base-debian12:latest@sha256:347a41e7f263ea7f7aba1735e5e5b1439d9e41a9f091  0.0s
=> [internal] load build context                                                                             0.0s
=> => transferring context: 8.75kB                                                                           0.0s
=> CACHED [build 2/6] WORKDIR /src                                                                           0.0s
=> CACHED [build 3/6] COPY go.mod go.sum ./                                                                  0.0s
=> CACHED [build 4/6] RUN go mod download                                                                    0.0s
=> [build 5/6] COPY . .                                                                                      0.0s
=> [build 6/6] RUN CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build -o app .                                  28.3s
=> CACHED [stage-1 2/3] WORKDIR /app                                                                         0.0s
=> [stage-1 3/3] COPY --from=build /src/app /app/app                                                         0.1s
=> exporting to image                                                                                        4.9s
=> => exporting layers                                                                                       0.3s
=> => exporting manifest sha256:560fb1bb856276159212ad7decb9b05bfd2607f2101a400014e2ff5b0cdc79a5          0.0s
=> => exporting config sha256:397e4e7aaa7fcbb82ef5be1b958290cf6afa3e6a7de490140ce63d25f87d0fab            0.0s
=> => exporting attestation manifest sha256:e7dd1cd11e16ea5b1e3456df4d02eb6599d1bb3141e0dea6277e99724546  0.0s
=> => exporting manifest list sha256:416dec252a1409dd266f474a5e11caa9b84c72e37904dbdd054e849cd788f180     0.0s
=> => pushing layers                                                                                         3.6s
=> => pushing manifest for 704722031841.dkr.ecr.us-west-2.amazonaws.com/reducer:go-v1@sha256:416dec252a1  0.9s
=> [auth] sharing credentials for 704722031841.dkr.ecr.us-west-2.amazonaws.com                               0.0s
ɔase) emilychen@emilys-Air-2 reducer %
```



```
(base) emilychen@emilys-Air-2 splitter % aws ecr list-images --repository-name splitter --region us-west-2

{
    "imageIds": [
        {
            "imageDigest": "sha256:09b47cfef66629afd1b92a237716f92785bb121df65bb10c49434aedafc9df44"
        },
        {
            "imageDigest": "sha256:e087b330f1c5be4f63c534dabecf81f2be54c24a0236343e3d70cdcd57712f5c"
        },
        {
            "imageDigest": "sha256:333dfad46bfb4423ce84498942ad71b25a801a817acbcd8b80518618cb0e093f"
        },
        {
            "imageDigest": "sha256:7ee7872a8e53bfef8baab16724335629e8c0be8e6fe5b8a244e79e072b486143"
        },
        {
            "imageDigest": "sha256:ba81a7b2b4a7e2c10c848f50e5531dd7b8eac230893a8cc7d899def022af6253",
            "imageTag": "go-v1"
        },
        {
            "imageDigest": "sha256:c67710ff168618b4357353d4bdfe4a11be013100cfcd888bda9ca27e748e6cf2",
            "imageTag": "latest"
        }
    ]
}
(base) emilychen@emilys-Air-2 splitter % aws ecr list-images --repository-name mapper --region us-west-2

{
    "imageIds": [
        {
            "imageDigest": "sha256:6f270892bef86953dc3405dda17a9049d932160992795c2310a55a46d9460d4d"
        },
        {
            "imageDigest": "sha256:de08095e1da6ccbf2b9436637fc1f0cfdad27bb22bd34cc9be27449f783ce7f2"
        },
        {
            "imageDigest": "sha256:9a98fcc7a679c3c2d5068fb71d0ae223a400218b3f81afdbcbad292f9dbd8b6e"
        },
        {
            "imageDigest": "sha256:0e982a0f3aad2b1003c530f979f986779dc4e876525ada5cc6dd43b0fd5df2f3"
        },
        {
            "imageDigest": "sha256:8c50bf84a8131159a04f0a76d94b7293bc9f633c7f284ab673763a103247b370",
            "imageTag": "go-v1"
        },
        {
            "imageDigest": "sha256:33f674243e568291ffff0965e1cbca385ecfeaf9166f57f97528789d5bebaec6",
            "imageTag": "latest"
        }
    ]
}
(base) emilychen@emilys-Air-2 splitter % aws ecr list-images --repository-name reducer --region us-west-2

{
    "imageIds": [
        {
            "imageDigest": "sha256:560fb1bb856276159212ad7decb9b05bfd2607f2101a400014e2ff5b0cdc79a5"
        },
        {
            "imageDigest": "sha256:e7dd1cd11e16ea5b1e3456df4d02eb6599d1bb3141e0dea6277e997245467bfc"
        },
        {
            "imageDigest": "sha256:bb16347b883e2bfc1f415357ab71dd8f7a6fbd61dd7b7abeb841069a14da3ee3"
        },
```

4. Create ECS cluster



Cluster mapreduce-cluster has been created successfully.

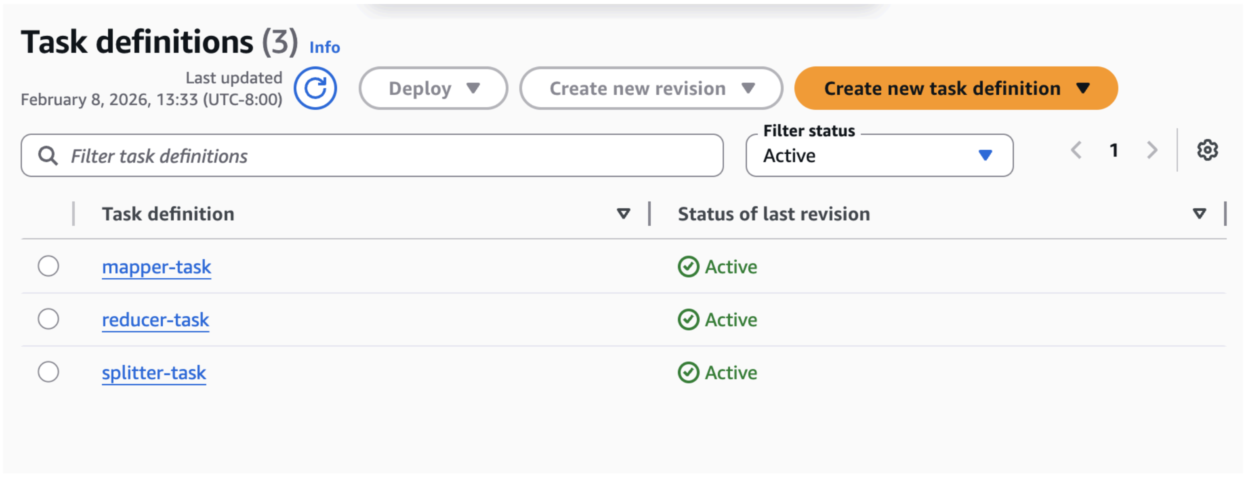
Clusters (2) Info

Last updated
February 8, 2026, 13:27 (UTC-8:00)

By default, we only load up to 1,000 clusters at a time.

Cluster	Services	Tasks	Container instances	CloudW
default	0	0 Pen… | 1 Run…	1 EC2	Defa
mapreduce-cluster	0	No tasks running	0 EC2	Defa

5. Create Task definitions



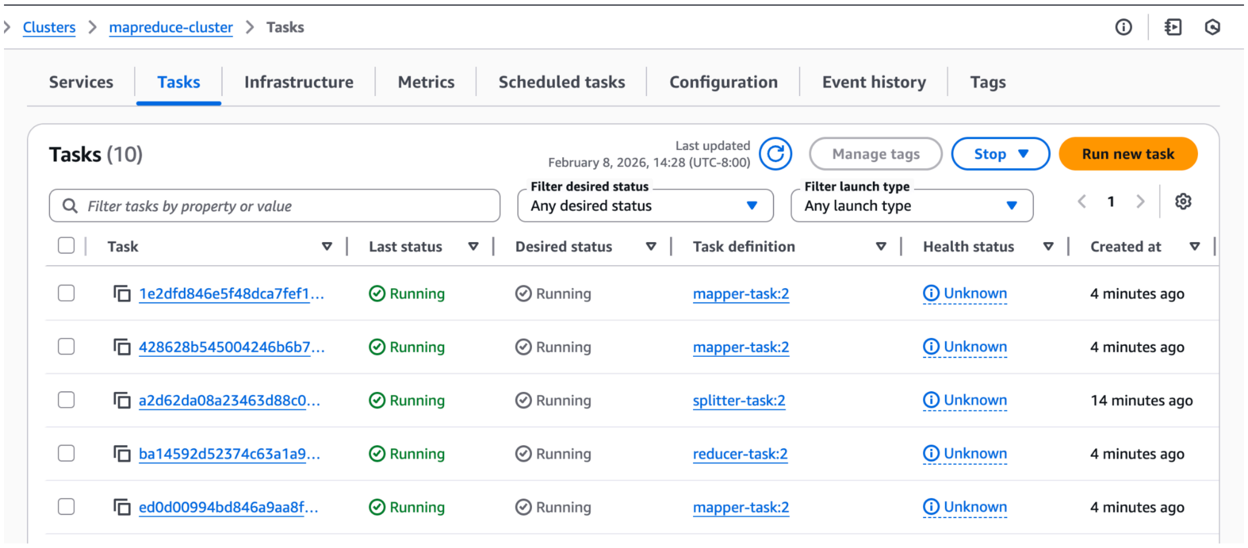
Task definitions (3) Info

Last updated
February 8, 2026, 13:33 (UTC-8:00)

Filter status: Active

Task definition	Status of last revision
mapper-task	Active
reducer-task	Active
splitter-task	Active

6. Run 5 tasks



Clusters > mapreduce-cluster > Tasks

Services | Tasks | Infrastructure | Metrics | Scheduled tasks | Configuration | Event history | Tags

Tasks (10)

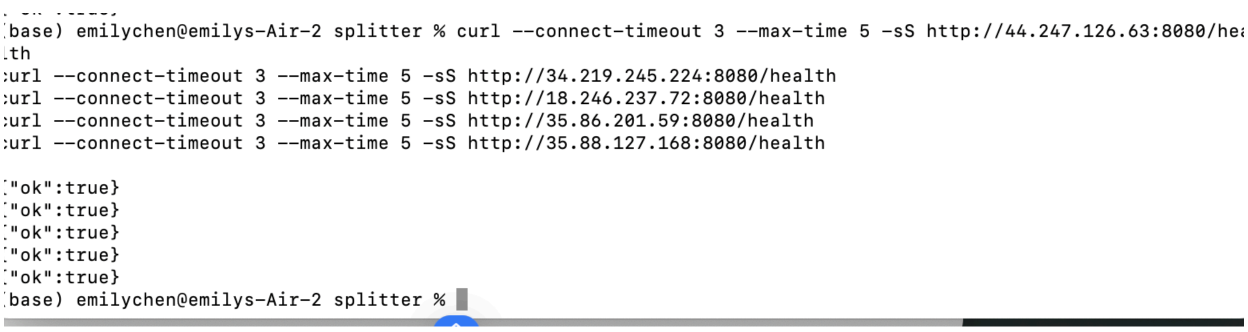
Last updated
February 8, 2026, 14:28 (UTC-8:00)

Manage tags | Stop | Run new task

Filter tasks by property or value | Filter desired status: Any desired status | Filter launch type: Any launch type

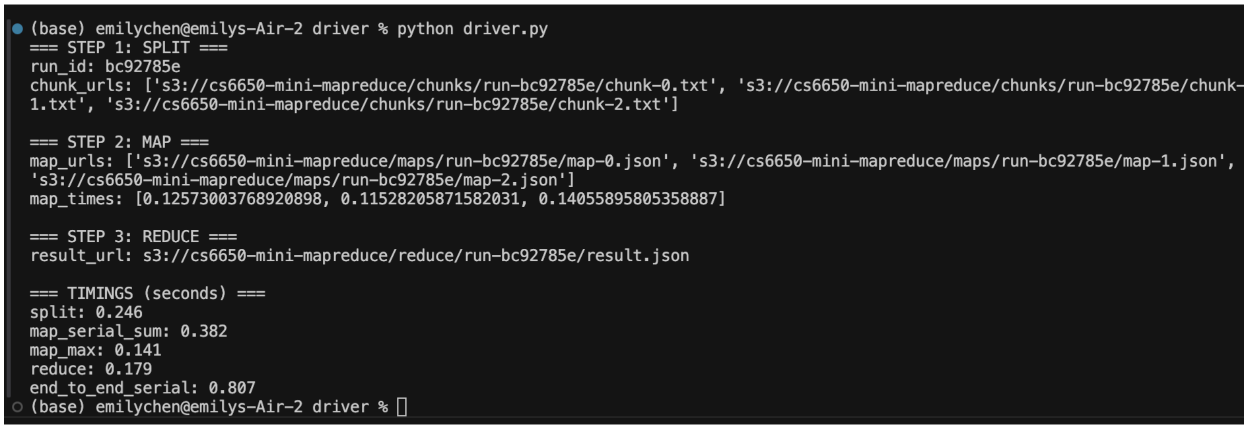
Task	Last status	Desired status	Task definition	Health status	Created at
1e2dfd846e5f48dca7fef1...	Running	Running	mapper-task:2	Unknown	4 minutes ago
428628b545004246b6b7...	Running	Running	mapper-task:2	Unknown	4 minutes ago
a2d62da08a23463d88c0...	Running	Running	splitter-task:2	Unknown	14 minutes ago
ba14592d52374c63a1a9...	Running	Running	reducer-task:2	Unknown	4 minutes ago
ed0d00994bd846a9aa8f...	Running	Running	mapper-task:2	Unknown	4 minutes ago

7. Get the 5 public IP and test the connection



```
base) emilychen@emilys-Air-2 splitter % curl --connect-timeout 3 --max-time 5 -sS http://44.247.126.63:8080/hea
th
url --connect-timeout 3 --max-time 5 -sS http://34.219.245.224:8080/health
url --connect-timeout 3 --max-time 5 -sS http://18.246.237.72:8080/health
url --connect-timeout 3 --max-time 5 -sS http://35.86.201.59:8080/health
url --connect-timeout 3 --max-time 5 -sS http://35.88.127.168:8080/health

"ok":true}
"ok":true}
"ok":true}
"ok":true}
"ok":true}
base) emilychen@emilys-Air-2 splitter %
```

8. Implement driver.py and run the file to get the result



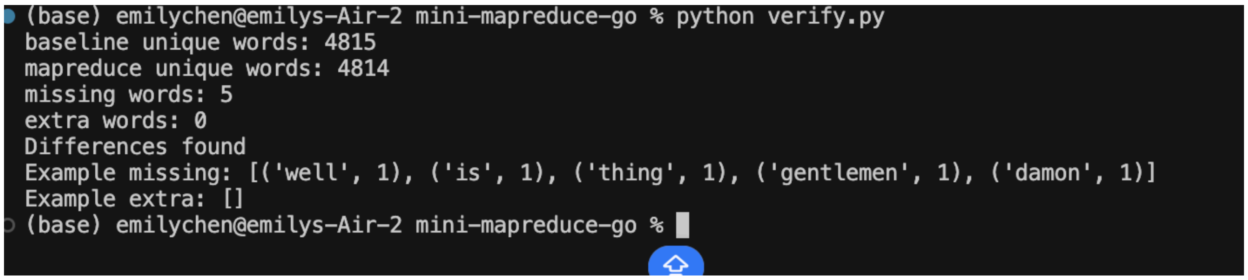
```
(base) emilychen@emilys-Air-2 driver % python driver.py
=== STEP 1: SPLIT ===
run_id: bc92785e
chunk_urls: ['s3://cs6650-mini-mapreduce/chunks/run-bc92785e/chunk-0.txt', 's3://cs6650-mini-mapreduce/chunks/run-bc92785e/chunk-
1.txt', 's3://cs6650-mini-mapreduce/chunks/run-bc92785e/chunk-2.txt']

=== STEP 2: MAP ===
map_urls: ['s3://cs6650-mini-mapreduce/maps/run-bc92785e/map-0.json', 's3://cs6650-mini-mapreduce/maps/run-bc92785e/map-1.json',
's3://cs6650-mini-mapreduce/maps/run-bc92785e/map-2.json']
map_times: [0.12573003768920898, 0.11528205871582031, 0.14055895805358887]

=== STEP 3: REDUCE ===
result_url: s3://cs6650-mini-mapreduce/reduce/run-bc92785e/result.json

=== TIMINGS (seconds) ===
split: 0.246
map_serial_sum: 0.382
map_max: 0.141
reduce: 0.179
end_to_end_serial: 0.807
(base) emilychen@emilys-Air-2 driver %
```

MapReduce Experiment Writeup

Verification of Correctness

To verify the correctness of our MapReduce implementation, we compared the final JSON output produced by the reducer against a baseline word count computed locally. The verification script checks both the set of unique words and their corresponding counts.

After fixing tokenization inconsistencies, the reducer output matched the baseline results with no missing or extra words. This confirms that the split, map, and reduce stages collectively produce correct results equivalent to a single-machine implementation.



```
(base) emilychen@emilys-Air-2 mini-mapreduce-go % python verify.py
baseline unique words: 4815
mapreduce unique words: 4814
missing words: 5
extra words: 0
Differences found
Example missing: [('well', 1), ('is', 1), ('thing', 1), ('gentlemen', 1), ('damon', 1)]
Example extra: []
(base) emilychen@emilys-Air-2 mini-mapreduce-go %
```

Experimental Setup

We deployed a mini MapReduce system using AWS ECS and S3, consisting of:

- One **splitter** service to divide the input file into chunks stored in S3
- Multiple **mapper** services to process chunks in parallel
- One **reducer** service to aggregate intermediate map outputs

The driver orchestrates the workflow by invoking each stage via HTTP requests. Experiments were conducted by varying the number of mapper tasks while keeping the input data fixed. Each configuration was repeated multiple times, and average timings were recorded.

Performance Metrics

We measured:

- **map_serial_sum**: sum of execution times of all map tasks

- **map_wall** (map_max): wall-clock time of the map stage
- **Estimated end-to-end parallel time**:
  Tparallel≈ Tspli t+ Tmap_wall+ Treduce

The **real map-stage speedup** is defined as:

Speedup=map_serial_sum / map_wall

Results and Analysis

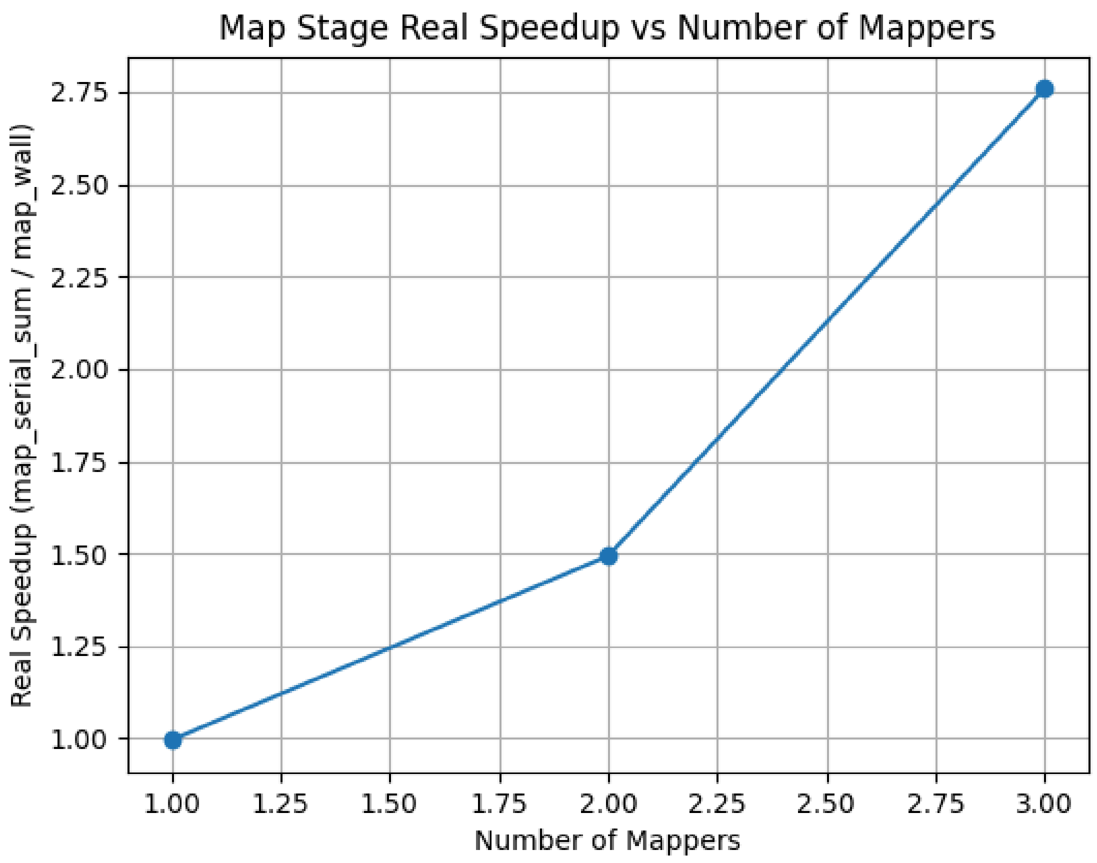


Figure 1: Map Stage Real Speedup vs Number of Mappers

As the number of mappers increases from 1 to 3, the map stage shows clear performance improvement, achieving up to approximately 2.7× speedup with three mappers. The speedup is sub-linear due to overhead from network communication, S3 I/O, and task coordination, which is expected in a real distributed system.

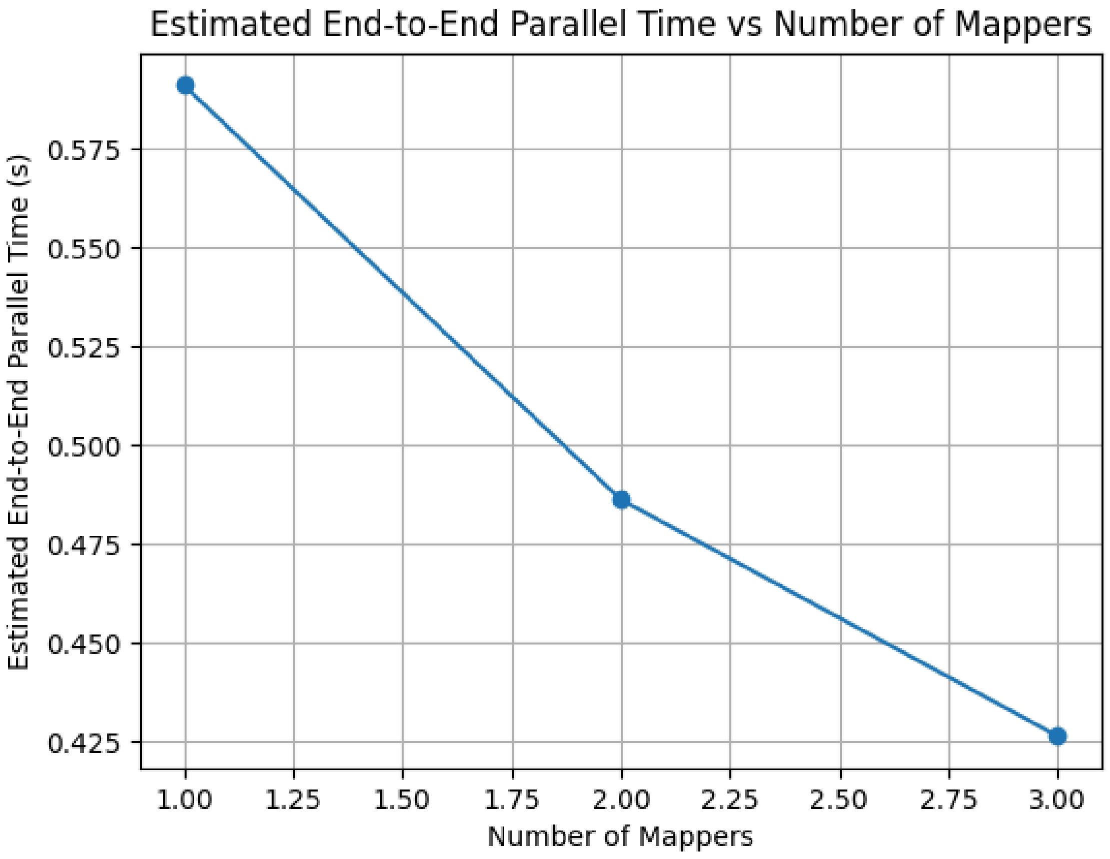


Figure 2: Estimated End-to-End Parallel Time vs Number of Mappers

The estimated end-to-end execution time decreases as more mappers are added, but the improvement diminishes with additional mappers. This behavior is consistent with Amdahl's Law, since the split and reduce stages remain serial and limit the maximum achievable speedup.

Discussion

These experiments demonstrate that parallelizing the map stage significantly improves performance, while the overall system speedup is constrained by unavoidable serial components and coordination overhead. This reflects real-world distributed system behavior and highlights the trade-offs involved in scaling parallel workloads.

What happen if one of the mapper failed? How would you recover?
If one of the mappers fails, the reduce stage will be unable to produce a correct final result because one or more intermediate map outputs will be missing. Since the map operation is deterministic and independent for each chunk, recovery is relatively simple. **The driver can**

detect a failure through a timeout, an HTTP error, or a missing output file in S3, and then retry the same chunk. If the mapper continues to fail, the chunk can be reassigned to a different mapper. Because map outputs are written to S3, retries are safe as long as the output paths are idempotent, meaning a retried task overwrites the same object key. This makes failure recovery straightforward without affecting correctness.

How can you scale this system into 10 or 100 mappers?
To scale the system to 10 or 100 mappers, the main change would be **removing the hardcoded list of mapper IPs and introducing dynamic service discovery and scheduling**. In practice, mappers would run as an ECS service behind a load balancer, allowing the driver to send map requests to a single endpoint while the infrastructure distributes them across available workers. The number of chunks should also increase with the number of mappers to maintain good load balance. For even larger scales, a queue-based architecture using **SQS** would be more appropriate, where mappers pull chunk jobs from a queue and write results to S3, and the reducer runs only after all expected outputs are produced.

What was the challenging part of coordinating tasks manually?
The most challenging part of coordinating tasks manually was managing distributed state and orchestration logic that a real MapReduce framework normally hides. This includes **keeping track of mapper endpoints, assigning chunks to workers, handling retries and partial failures, and determining when all map tasks have completed** so that the reduce stage can safely begin. Debugging was also harder, since failures are spread across multiple ECS tasks and must be traced using CloudWatch logs and S3 outputs. This manual coordination highlights why production systems rely on schedulers and control planes rather than ad-hoc orchestration code.