Fixing ALB 502 for ASG-hosted FastAPI app — Quick Runbook

What worked (repeatably) to turn 502s into 200s, and how to make it stick.

Symptoms

- ALB /health returned **502**.
- Target group showed instances in *draining* and *unhealthy*.
- SSM session showed Docker/app not running on new instances.

Likely causes hit

- User data didn't start the app with the correct **module path** or didn't bind to 0.0.0.0.
- Target group misaligned (wrong port/path/matcher) during bring■up.
- Instance refresh left old instances *draining* while new ones were still *unhealthy*.

Prereqs (handy env vars)

```
export REGION=us-east-1
export ASG_NAME=$(terraform -chdir=infra/envs/dev output -raw asg_name)
export ALB_DNS=$(terraform -chdir=infra/envs/dev output -raw alb_dns)
export TG_ARN=$(terraform -chdir=infra/envs/dev output -raw target_group_arn)
```

1) Verify ALB/TG settings

```
aws elbv2 describe-target-groups --region "$REGION" --target-group-arns "$TG_ARN" \
  --query 'TargetGroups[0].{Proto:Protocol,Port:Port,Path:HealthCheckPath,Matcher:Matcher.HttpCode}' -

# Expect: Port=8080, Protocol=HTTP, Path=/health, Matcher=200-399 (during bring-up)
curl -sS -X GET -i "http://$ALB_DNS/health"  # Expect HTTP/200
```

2) Fix an unhealthy instance (via SSM) and prove it returns 200

```
# newest instance
INSTANCE_ID=$(aws autoscaling describe-auto-scaling-groups \
  --auto-scaling-group-names "$ASG_NAME" \
  --query 'AutoScalingGroups[0].Instances[*].[LaunchTime,InstanceId]' --output text | sort | tail -n1
aws ssm start-session --target "$INSTANCE_ID" --region "$REGION"

# On the box — if Docker/app isn't up, install/start and run the app explicitly:
sudo dnf -y install docker awscli amazon-ssm-agent || true
sudo systemctl enable --now docker amazon-ssm-agent

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text --region "$REGION")
ECR_URI="${ACCOUNT_ID}.dkr.ecr.${REGION}.amazonaws.com/aws-ha-webapp-app"
TAG=$(aws ecr describe-images --region "$REGION" --repository-name aws-ha-webapp-app \
  --query 'reverse(sort_by(imageDetails,&imagePushedAt))[0].imageTags[0]' --output text)

aws ecr get-login-password --region "$REGION" | sudo docker login --username AWS --password-stdin "$AC
```

```
sudo docker rm -f app || true
sudo docker pull "$ECR_URI:$TAG"
sudo docker run -d --name app -p 8080:8080 --restart unless-stopped -e PORT=8080 \
  "$ECR_URI:$TAG" uvicorn src.app.main:app --host 0.0.0.0 --port 8080

curl -i http://127.0.0.1:8080/health   # Expect HTTP/200
```

3) Make it stick (user_data + ASG health)

Update launch template user_data to run Uvicorn with the correct module path and 0.0.0.0 binding:

```
# infra/modules/asg/user_data.tpl (excerpt)
/usr/bin/docker run -d --name app \
  -p "${APP_PORT}:${APP_PORT}" --restart unless-stopped \
  -e PORT="${APP_PORT}" \
  "${ECR_REPO_URI}:${IMAGE_TAG}" \
  uvicorn src.app.main:app --host 0.0.0.0 --port "${APP_PORT}"

# infra/modules/asg/main.tf
health_check_type         = "ELB"
health_check_grace_period = 120
```

4) Kick an Instance Refresh & watch target health turn healthy

```
aws autoscaling start-instance-refresh --auto-scaling-group-name "$ASG_NAME" --strategy Rolling
# if already in progress
aws autoscaling cancel-instance-refresh --auto-scaling-group-name "$ASG_NAME"

watch -n 5 'aws elbv2 describe-target-health --target-group-arn "$TG_ARN" \
  --query "TargetHealthDescriptions[].{Id:Target.Id,State:TargetHealth.State,Reason:TargetHealth.Reasc
```

5) Understand draining vs unhealthy

- **draining**: instance is being removed; it will go away after TargetGroup deregistration delay.
- **unhealthy**: instance still registered, health check failing (fix app, path, port, or SG).

6) Sanity checks

```
# App responds through ALB
curl -sS -X GET -i "http://$ALB_DNS/health"

# Target healthy
aws elbv2 describe-target-health --target-group-arn "$TG_ARN" \
  --query 'TargetHealthDescriptions[].{Id:Target.Id,State:TargetHealth.State}' --output table

# Security groups (ALB 80/443 from world, app 8080 from ALB SG, egress 0.0.0.0/0)
ALB_SG=$(terraform -chdir=infra/envs/dev output -raw alb_sg_id)
APP_SG=$(terraform -chdir=infra/envs/dev output -raw app_sg_id)
aws ec2 describe-security-groups --group-ids "$ALB_SG" "$APP_SG" --output table
```

What changed in code

- User data runs the app with explicit entrypoint: uvicorn src.app.main:app --host 0.0.0.0 --port ${APP_PORT}.

- ASG health check uses ELB with a 120s grace period.
- ALB Target Group: HTTP:8080, health check path /health, matcher 200-399.

Gotchas noted

- If SSM says *TargetNotConnected*: ensure interface endpoints + SSM agent + EC2 role perms.
- If Docker can’t pull: add ecr:GetAuthorizationToken, ecr:BatchGetImage, ecr:DescribeImages, ecr:GetDownloadUrlForLayer to the EC2 role.
- If ALB still 502s: confirm app binds to 0.0.0.0, not 127.0.0.1.