Here is the final, step-by-step summary to securely handle SMTP credentials in your GitHub flow.

### The Architecture

1. **GitHub Secrets:** Store the raw password.
2. **Bash Script:** Reads the secret from GitHub Env $\to$ Pushes to AWS Secrets Manager.
3. **Terraform:** Reads from AWS Secrets Manager $\to$ Injects into ECS Container.

### Step 1: Secure the Credentials (GitHub)

Go to your GitHub Repository $\to$ Settings $\to$ Secrets and variables $\to$ Actions.

Create these two Repository Secrets:

* GRAFANA\_SMTP\_USER $\to$ shilpakangya2025@gmail.com
* GRAFANA\_SMTP\_PASSWORD $\to$ *(Your Google App Password)*

### Step 2: Grant Permissions (AWS IAM)

The IAM Role that GitHub Actions uses (AWS\_ROLE\_ARN) needs permission to write this secret.

* **Where:** AWS Console $\to$ IAM $\to$ Roles $\to$ [Your GitHub Action Role].
* **Action:** Add an Inline Policy.

JSON

{  
 "Version": "2012-10-17",  
 "Statement": [  
 {  
 "Sid": "AllowManageGrafanaSecret",  
 "Effect": "Allow",  
 "Action": [  
 "secretsmanager:DescribeSecret",  
 "secretsmanager:GetSecretValue",  
 "secretsmanager:CreateSecret",  
 "secretsmanager:PutSecretValue",  
 "secretsmanager:UpdateSecret"  
 ],  
 "Resource": "arn:aws:secretsmanager:us-east-1:\*:secret:grafana/smtp\*"  
 }  
 ]  
}

### Step 3: Create the Script

Create a file at scripts/ensure-grafana-smtp-secret.sh in your repo.

* *Make sure to run chmod +x scripts/ensure-grafana-smtp-secret.sh locally before committing.*

Bash

#!/bin/bash  
# scripts/ensure-grafana-smtp-secret.sh  
  
if [[ -z "$SMTP\_USER" || -z "$SMTP\_PASSWORD" ]]; then  
 echo "Error: Env variables missing."  
 exit 1  
fi  
  
SECRET\_NAME="grafana/smtp"  
REGION="us-east-1"  
  
# Construct JSON securely  
SECRET\_STRING=$(jq -n \  
 --arg user "$SMTP\_USER" \  
 --arg password "$SMTP\_PASSWORD" \  
 --arg host "smtp.gmail.com:587" \  
 --arg from "$SMTP\_USER" \  
 --arg name "IoT Factory Simulator" \  
 '{  
 SMTP\_USER: $user,  
 SMTP\_PASSWORD: $password,  
 SMTP\_HOST: $host,  
 SMTP\_FROM: $from,  
 SMTP\_NAME: $name  
 }')  
  
# Create or Update Secret  
aws secretsmanager describe-secret --secret-id "$SECRET\_NAME" --region "$REGION" >/dev/null 2>&1  
  
if [ $? -eq 0 ]; then  
 echo "Updating secret $SECRET\_NAME..."  
 aws secretsmanager put-secret-value --secret-id "$SECRET\_NAME" --secret-string "$SECRET\_STRING" --region "$REGION"  
else  
 echo "Creating secret $SECRET\_NAME..."  
 aws secretsmanager create-secret --name "$SECRET\_NAME" --secret-string "$SECRET\_STRING" --region "$REGION"  
fi

### Step 4: Update the Workflow

Update .github/workflows/deploy.yml. Add the "Ensure Secret" step **before** Terraform Setup in **both** plan-dev and plan-prod.

YAML

# ... (After Configure AWS Credentials) ...  
  
 - name: Ensure Grafana SMTP Secret  
 env:  
 SMTP\_USER: ${{ secrets.GRAFANA\_SMTP\_USER }}  
 SMTP\_PASSWORD: ${{ secrets.GRAFANA\_SMTP\_PASSWORD }}  
 run: |  
 chmod +x scripts/ensure-grafana-smtp-secret.sh  
 ./scripts/ensure-grafana-smtp-secret.sh  
  
 # ... (Then Setup Terraform, Inject Certs, and Run Plan) ...

### How to Verify

1. **Run the Pipeline:** Push code or merge a PR.
2. **Check AWS:** Go to AWS Secrets Manager. You should see grafana/smtp created or updated.
3. **Check Deployment:** Terraform will deploy ECS. Your ECS Execution Role (already configured in your terraform\_v5.tar) will pull this secret, and Grafana will start with email capabilities.