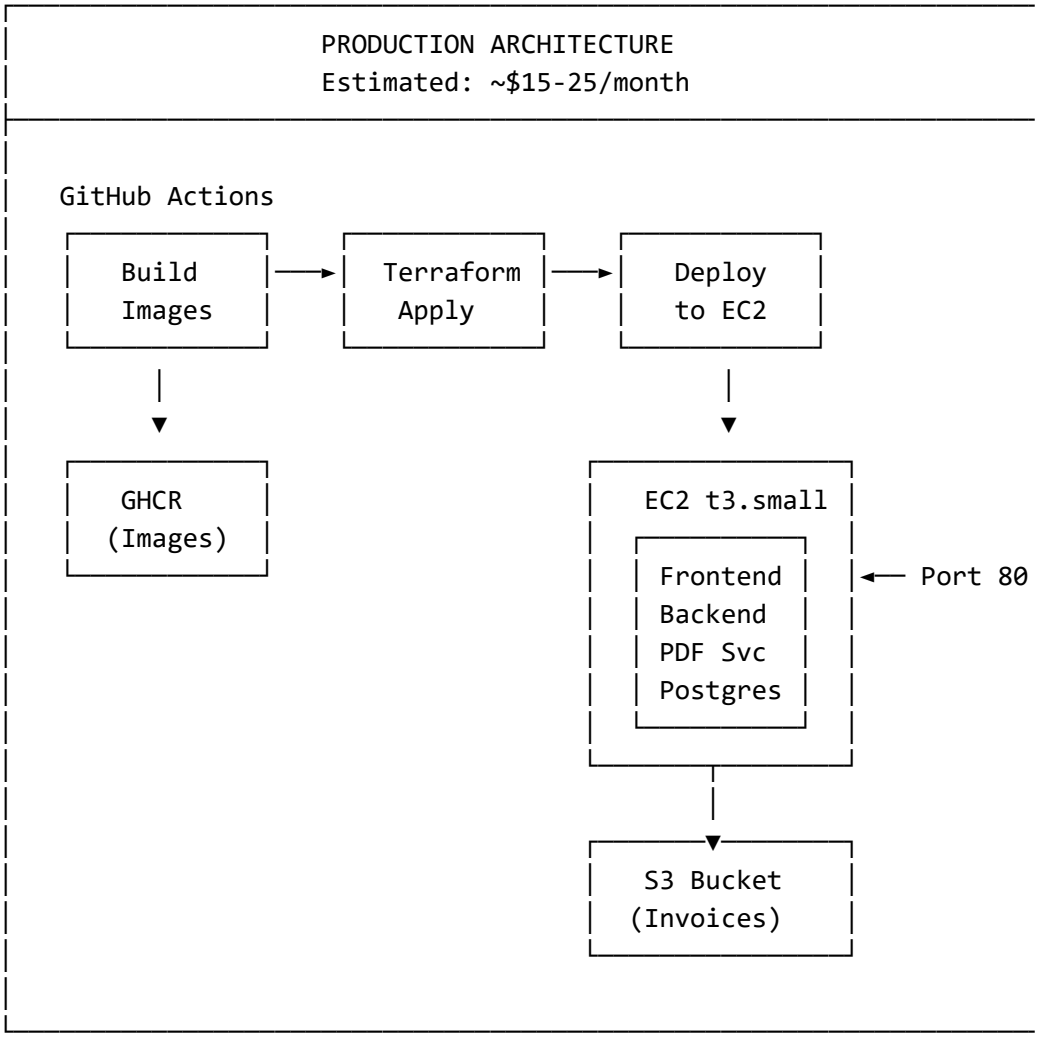
SyncLedger - AWS Infrastructure & CI/CD Deployment Guide

Architecture Overview



Cost Breakdown

Resource	Spec	Monthly Cost
EC2	t3.small (2 vCPU, 2GB)	~$15
EBS	20GB gp3	~$1.60
Elastic IP	Attached to instance	FREE
S3	< 5GB standard	~$0.15

CloudWatch Logs	5GB	~$2.50
SSM Parameters	Standard	FREE
GitHub Actions	2000 min/month free	FREE
GHCR	500MB free storage	FREE
Total		**~$20-25/month**
With Free Tier	First 12 months	**~$3-8/month**

Step-by-Step Setup

Prerequisites

- AWS Account (with free tier ideally)
- GitHub repository for SyncLedger
- Terraform >= 1.5 installed locally
- AWS CLI v2 installed locally

Step 1: Bootstrap AWS (One-Time)

This creates the Terraform state bucket and DynamoDB lock table using your AWS root/admin account.

```
# Configure AWS CLI with your admin credentials
aws configure
# Region: us-east-1
# Output: json

# Bootstrap Terraform state storage
cd terraform/bootstrap
terraform init
terraform apply
```

Save the outputs - you'll need the S3 bucket name.

Step 2: Create EC2 Key Pair (Optional, for SSH)

```
# Create key pair for SSH access
aws ec2 create-key-pair \
  --key-name syncledger-key \
  --query 'KeyMaterial' \
  --output text > syncledger-key.pem

chmod 400 syncledger-key.pem
```

Step 3: Configure Terraform Variables

```
cd terraform

# Copy example file
cp terraform.tfvars.example terraform.tfvars

# Edit with your values
```

terraform.tfvars:

```
aws_region    = "us-east-1"
environment   = "prod"
project_name  = "syncledger"
instance_type = "t3.small"     # or t3.micro for free tier
ec2_key_name  = "syncledger-key"  # from step 2

# These are sensitive - use strong values!
# db_password and jwt_secret are passed via CLI or environment varia
```

Step 4: Deploy Infrastructure

```
cd terraform

# Enable remote state (optional but recommended)
# Uncomment the backend "s3" block in versions.tf

# Initialize Terraform
terraform init

# Preview changes
terraform plan \
  -var="db_password=YOUR_STRONG_DB_PASSWORD" \
  -var="jwt_secret=$(openssl rand -base64 48)"

# Apply
terraform apply \
  -var="db_password=YOUR_STRONG_DB_PASSWORD" \
  -var="jwt_secret=$(openssl rand -base64 48)"
```

Save the outputs, especially:

- `deployer_access_key_id`
- `deployer_secret_access_key` (run `terraform output -raw deployer_secret_access_key`)
- `app_public_ip`

Step 5: Configure GitHub Repository Secrets

Go to your GitHub repo → Settings → Secrets and variables → Actions

Required Secrets:

Secret Name	Value	Source
`AWS_ACCESS_KEY_ID`	Deployer access key ID	Terraform output
`AWS_SECRET_ACCESS_KEY`	Deployer secret key	Terraform output
`DB_PASSWORD`	Database password	Your chosen password
`JWT_SECRET`	JWT signing key	Generated in step 4

Required Variables (Settings → Variables):

Variable Name	Value
`AWS_REGION`	`us-east-1`
`EC2_KEY_NAME`	`syncledger-key`
`DOMAIN_NAME`	(leave empty if no domain)

Step 6: Build & Deploy

```
# Push code to main branch - this triggers the Build workflow
git add .
git commit -m "Add infrastructure and CI/CD"
git push origin main
```

Then go to GitHub Actions and:

1. Wait for **Build & Push Images** to complete
2. Run **Deploy** workflow manually (Actions → Deploy → Run workflow)
3. Select `deploy` action and `prod` environment

Step 7: Verify Deployment

```
# Get the public IP from Terraform output
IP=$(cd terraform && terraform output -raw app_public_ip)

# Check health
curl http://$IP/health
curl http://$IP/api/actuator/health

# Open in browser
echo "Open: http://$IP"
```

GitHub Actions Workflows

1. Build & Push Images (`build.yml`)

- **Trigger:** Push to `main` (auto-detects changed services)
- **Manual:** Can rebuild specific services
- **Builds:** Frontend, Backend, PDF Service → GHCR
- **Cost:** Free (within 2000 min/month)

2. Deploy (`deploy.yml`)

- **Trigger:** Manual only (workflow_dispatch)
- **Actions:** `plan-only` or `deploy`
- **Process:** Terraform plan → apply → pull images on EC2 → health check

3. Destroy (`destroy.yml`)

- **Trigger:** Manual only with "DESTROY" confirmation

- **Process:** Stops containers → Terraform destroy → removes all AWS resources
- **Safety:** Requires typing "DESTROY" + environment protection approval

---

IAM Security Design

Deployer User (`syncledger-prod-deployer`)

Located under IAM path `/syncledger/` with a custom policy that restricts access to:

Resource	Access Level	Scope
EC2 Instances	Create/Manage/Terminate	Region-locked
Security Groups	Full	Default VPC only
Elastic IPs	Full	Region-locked
S3	Full	`syncledger-*` buckets only
IAM Roles	Create/Manage	`syncledger-*` roles only
CloudWatch Logs	Create/Write	`/syncledger/*` groups only
SSM Parameters	Read/Write	`/syncledger/*` params only
VPC/Subnet	Read-only	Default VPC

The deployer user **cannot**:

- Access other projects' resources
- Create IAM users or modify policies outside the project
- Access resources in other regions
- Read/modify resources without the `syncledger-` prefix

Destroying Resources

When you no longer need the project running:

Option A: Via GitHub Actions (Recommended)

1. Go to Actions → **Destroy Infrastructure**
2. Select environment → Type `DESTROY` → Run
3. All resources will be removed, monthly charges stop

Option B: Via CLI

```
cd terraform
terraform destroy \
  -var="db_password=any" \
  -var="jwt_secret=any"
```

Troubleshooting

SSH into EC2

```
ssh -i syncledger-key.pem ec2-user@<PUBLIC_IP>

# Check containers
cd /opt/syncledger
docker-compose -f docker-compose.prod.yml ps
docker-compose -f docker-compose.prod.yml logs -f

# Check setup log
cat /var/log/syncledger-setup.log
```

Common Issues

Issue	Fix
Health check fails	Wait 2-3 min for Java to start. Check: `docker logs syncledger-backend`
Images not pulling	Ensure GHCR packages are public, or configure docker login on EC2
DB connection error	Check DB_PASSWORD matches in `.env` and Terraform vars
Port 80 not accessible	Verify security group allows HTTP from 0.0.0.0/0
Terraform state locked	Delete lock: `aws dynamodb delete-item --table-name syncledger-terraform-locks --key '{"LockID":{"S":"..."}}'`

Scaling Up

When traffic grows, consider:

1. **t3.small** → **t3.medium** ($30/mo): More RAM for Java

2. **Add RDS PostgreSQL** ($15/mo): Separate DB from EC2
3. **Add CloudFront CDN** ($5/mo): Global frontend caching
4. **Add Application Load Balancer** ($20/mo): SSL termination, health checks