AWS Docker Deploy — Key Points & Checklist

Concise, actionable checklist of what mattered in our session. Keep this as your quick reference.

---

1) Big picture

- EC2 acts as a **VPS**. Same workflow as any VPS: SSH → install Docker → pull code → run container.
- Use **EC2 (t2/t3.micro)** for learning; free-tier / minimal cost.

---

2) Launch EC2 (required values)

- AMI: **Ubuntu Server 22.04 LTS (64-bit)**
- Instance type: **t2.micro** or **t3.micro** (Free-tier eligible)
- Storage: **gp3**, **8 GB** (default fine)
- Key pair: create and download `docker-ec2-key.pem` (keep safe)
- Network: Auto-assign public IPv4 = **Enable**

---

3) Security Group (essential rules)

- Only these inbound rules while developing:
- **SSH 22** → initially `0.0.0.0/0` for CI runners (temporary). For manual access, restrict to your IP `x.x.x.x/32`.
- **Custom TCP** → **3000** (or your app port) → `0.0.0.0/0`
- Add any other app port you actually use (e.g., **8000**) if container maps to it.

Do not open unused ports (80/443) unless needed.

---

4) SSH key rules & command

- Set permissions once locally:

```
chmod 400 /path/to/docker-ec2-key.pem
```

- SSH command:

```
ssh -i /path/to/docker-ec2-key.pem ubuntu@<EC2_PUBLIC_IPV4>
```

- Recommended: move key to `~/.ssh/` and use full path.

5) Server bootstrap (run on EC2 after SSH)

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y ca-certificates curl gnupg git
curl -fsSL https://get.docker.com | sudo sh
sudo usermod -aG docker ubuntu
exit   # reconnect to apply group
ssh -i key.pem ubuntu@<IP>
docker --version
docker compose version  # plugin or v2
```

---

6) Repo clone (on EC2)

- Use HTTPS clone on server (no SSH key configured on EC2):

```
git clone https://github.com/<user>/<repo>.git
cd <repo>
ls
```

---

7) Docker run & sanity checks (on EC2)

- Ensure `docker-compose.yml` has port mapping, e.g.:

```
ports:
  - "3000:3000"
```

- Build and run:

```
docker compose up -d --build
# or if action logs freeze, split build and up:
docker compose build --no-cache
docker compose up -d
```

- Test internally on EC2:

```
curl http://localhost:3000
```

- Test externally from your laptop:

```
curl http://<EC2_PUBLIC_IPV4>:3000
```

If `curl` from laptop hangs → security group blocking port.

---

8) Common troubleshooting

- `git clone git@github.com:...` fails on EC2 → use HTTPS or add SSH key to EC2 GitHub account.
- `curl` hangs (no response) → check Security Group inbound rules for that port.
- `dial tcp <ip>:22: i/o timeout` in Actions → GitHub runner cannot reach SSH port; open SSH (0.0.0.0/0) or use SSM/bastion.
- GitHub Actions appears stuck during `docker compose up --build` → build can be slow; split build and up, increase `command_timeout`, and enable debug.

---

9) CI/CD (GitHub Actions) — correct minimal approach

- Use SSH-based deploy (appleboy/ssh-action). Secrets required:
- `EC2_HOST` → public IPv4
- `EC2_USER` → `ubuntu`
- `EC2_SSH_KEY` → full contents of `.pem`
- `EC2_PATH` → path on EC2 (e.g. `/home/ubuntu/projects/docker-action-aws`)

Minimal example (use this YAML):

```
name: Deploy Node App to EC2
on:
  push:
    branches: [ main ]
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Deploy via SSH
        uses: appleboy/ssh-action@v1.0.3
        with:
          host: ${{ secrets.EC2_HOST }}
          username: ${{ secrets.EC2_USER }}
          key: ${{ secrets.EC2_SSH_KEY }}
          timeout: 60s
          command_timeout: 15m
          debug: true
          script: |
```

```
            set -e
            cd ${{ secrets.EC2_PATH }}
            git pull origin main
            docker compose build --no-cache
            docker compose up -d
```

Notes: - Use `docker compose build` and then `up -d` separately so Action logs stream. - If Actions cannot SSH, set SSH inbound to `0.0.0.0/0` temporarily.

---

10) Costs & lifecycle

- **Stop** instance to pause compute costs (disk persists). Do **not** terminate if you want to reuse.
- Public IPv4 changes on stop/start unless you allocate an Elastic IP.
- EBS gp3 volume storage cost is minimal for 8 GB — keep it if you want to reuse the server.
- Detaching the volume will break the instance (do not detach unless you plan manual reattach).

---

11) Elastic Beanstalk / S3 cleanup

- `elasticbeanstalk-<region>-<account>` buckets store EB uploads; if you terminated EB environment and do not plan to reuse EB, delete the bucket to avoid leftover artifacts.

---

12) Security hardening (next session)

- Restrict SSH to your IP when CI is not running from Actions, or implement GitHub Actions runner allowlist / use AWS SSM Session Manager or bastion host.
- Consider using Elastic IP if you want static IP for DNS and GitHub secret stability.
- Use snapshots for backups before destructive actions.

---

13) Short command appendix

- SSH:

```
ssh -i ~/.ssh/docker-ec2-key.pem ubuntu@<IP>
```

- Set key perms:

```
chmod 400 docker-ec2-key.pem
```

- Docker checks:

```
docker --version
docker compose version
docker ps
docker compose logs --tail=50
```

- Stop instance (save cost): AWS Console → Instances → Stop

KEEP THIS: focused checklist — follow these steps verbatim when you resume. Good progress.