**1️⃣ Clone & Setup**

git clone <your-repo-url>

cd saffaran-aviation-automation

Make sure you have these tools installed:

* **AWS CLI v2** (configured with IAM user/role with MFA + admin rights)
* **kubectl** (for EKS)
* **terraform** (>=1.6)
* **docker** (for local builds)
* **redis-cli** (optional, for Redis check)
* **jenkins** (for CI/CD pipeline, or use Jenkins EC2 setup provided)

**2️⃣ Provision AWS Infrastructure (Terraform)**

cd terraform

terraform init

terraform plan

terraform apply -auto-approve

✅ This creates:

* VPC, Subnets, Security Groups
* EKS Cluster
* Redis (ElastiCache)
* S3 Bucket
* IAM Roles + Policies
* CloudWatch + CloudTrail

**3️⃣ Deploy Application (Kubernetes Manifests)**

Update kubeconfig:

aws eks update-kubeconfig --region us-east-1 --name saffaran-eks-cluster

Deploy app:

cd ../k8s

kubectl apply -f namespace.yaml

kubectl apply -f secrets.yaml

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

kubectl apply -f ingress.yaml

kubectl apply -f hpa.yaml

Check:

kubectl get pods -n saffaran

kubectl get svc -n saffaran

**4️⃣ Build & Push Docker Image (Local or CI)**

If you want to build manually:

cd ../app

docker build -t saffaran-api .

docker tag saffaran-api:latest <your-ecr-repo-uri>:latest

docker push <your-ecr-repo-uri>:latest

If you want Jenkins to handle it:

* Push repo to **GitHub**.
* Point Jenkins pipeline to it.
* Jenkinsfile will handle build + push + deploy automatically.

**5️⃣ Test the Application**

Port forward to API:

kubectl port-forward -n saffaran svc/saffaran-api 8080:80

Health check:

curl http://127.0.0.1:8080/health

Send sample turbine data:

curl -X POST http://127.0.0.1:8080/report \

-H "Content-Type: application/json" \

-d @sample\_data/turbine\_sample.json

Expected result:

{"status":"accepted","object\_key":"reports/YYYY/MM/DD/uuid.json"}

**6️⃣ Verify Storage & Logs**

* **S3**: aws s3 ls <bucket-name>/reports/ --recursive
* **Redis**:
* redis-cli -h <redis-endpoint> HGETALL reports
* **CloudWatch Logs**: check /aws/saffaran/app

**7️⃣ Run Verification Script**

If you downloaded **saffaran-verification.zip**:

unzip saffaran-verification.zip

cd saffaran-verification

./verify.sh

This runs all the checks for you automatically ✅

**🔒 Security & Best Practices**

* Always use **IAM roles with MFA** for Terraform/Jenkins.
* Enable **encryption (KMS)** for S3 & Redis.
* Restrict **security groups** to allow only required traffic.
* Monitor with **CloudWatch alarms**.

⚡ That’s the full process:

1. Terraform infra → 2. Kubernetes deploy → 3. Docker image → 4. Jenkins CI/CD → 5. Test API → 6. Verify with S3/Redis/Logs.

👉 Do you want me to **merge the verification script inside the repo** (so Jenkins can run it automatically after deploy), or keep it as a separate package?

**You said:**

everything should be on ec2 instance or local machine?to be installed

**ChatGPT said:**

Excellent point 🚀 — deciding **where** to install and run each piece is key for production vs. development. Here’s a breakdown:

**⚙️ Deployment Location Breakdown**

**✅ Run on Local Machine (Dev/Testing Only)**

These tools are usually installed **locally** for development, testing, or initial provisioning:

* **Terraform** → You run it locally to create AWS resources (or from Jenkins pipeline).
* **AWS CLI** → For local validation, S3/Redis checks, kubeconfig setup.
* **kubectl** → To interact with EKS cluster after it’s created.
* **docker** → Build/test containers locally before pushing to ECR.

👉 Local machine = developer workstation.  
👉 In production, Terraform/Jenkins usually handle these instead of devs running them manually.

**✅ Run on EC2 Instance(s) (Production/Automation)**

These are meant to live on EC2 (or AWS managed services):

* **Jenkins** → Runs on an EC2 instance (automation server for CI/CD).
* **Kubernetes workloads** → Run on EC2 worker nodes **inside EKS cluster** (not on your local machine).
* **Redis (ElastiCache)** → AWS-managed, lives inside your **VPC**, not local.
* **S3, CloudWatch, CloudTrail, IAM, VPC** → All are AWS managed services.

**🔒 Typical Production Setup**

1. **Dev Machine (your laptop/PC)**
   * Has Terraform, AWS CLI, kubectl, docker → for initial setup and testing.
   * Pushes code to GitHub.
2. **Jenkins EC2 Instance**
   * Jenkins server pulls repo from GitHub.
   * Runs Terraform (IaC), docker builds, kubectl apply.
   * Handles deployment pipelines.
3. **EKS Cluster (EC2 worker nodes or Fargate)**
   * Runs your **Flask API containers**.
   * Uses Kubernetes manifests from repo.
4. **AWS Managed Services**
   * **S3** stores turbine reports.
   * **ElastiCache Redis** caches results.
   * **CloudWatch** collects logs/metrics.
   * **CloudTrail** audits activity.
   * **IAM + MFA** secures everything.

**🔑 Rule of Thumb**

* **Local machine** = For developers (setup, debugging, initial deploy).
* **EC2/Jenkins/EKS** = For continuous deployment + running workloads.
* **AWS Managed Services** = For storage, monitoring, caching, logging.

👉 Do you want me to make a **deployment diagram (architecture map)** that visually shows what runs **locally**, what runs on **EC2**, and what’s **AWS managed**? That way you’ll have a clear big picture.

**You said:**

yes

**ChatGPT said:**