Here’s a production-ready Docker + Kubernetes setup for your SIGAM app, with daily DB backups and minimal code changes to support environment-based DB config.

What I added

* Dockerfiles
  + server/Dockerfile
  + client/Dockerfile
  + server/.dockerignore
  + client/.dockerignore
  + client/nginx.conf
* Kubernetes
  + infra/k8s/namespace.yaml
  + infra/k8s/db-secret.yaml
  + infra/k8s/postgres-statefulset.yaml
  + infra/k8s/postgres-backup-pvc.yaml
  + infra/k8s/postgres-backup-cronjob.yaml
  + infra/k8s/server-deployment.yaml (with Prisma migrations as initContainer)
  + infra/k8s/client-deployment.yaml
  + infra/k8s/ingress.yaml
* Docs
  + infra/README.md: build, push, deploy, and restore instructions
* Server fix
  + server/src/app.module.ts: TypeORM now reads DB config from env (no more localhost hardcode)

Key details

* Stack detection: Prisma uses PostgreSQL; server runs NestJS on port 3001; client is Vite React built to static assets and served by Nginx.
* Migrations: Server Deployment runs an initContainer with npx prisma migrate deploy (I bundled Prisma CLI into the final image for this).
* Backups: A CronJob runs daily at midnight, dumps the database to a PVC, and keeps the last 7 days.
* Ingress: Two hostnames out of the box: sigam.example.com (client) and api.sigam.example.com (server). Adjust for your domain/TLS.

What you need to change before deploy

* Set real registry image names:
  + infra/k8s/server-deployment.yaml: image: sigam/server:latest → your registry
  + infra/k8s/client-deployment.yaml: image: sigam/client:latest → your registry
* Set DB credentials and URL:
  + infra/k8s/db-secret.yaml: POSTGRES\_PASSWORD and DATABASE\_URL
* Set client API URL at build time:
  + client image build arg: --build-arg VITE\_API\_URL=[https://api.yourdomain](https://api.yourdomain/)
* Set Ingress hosts:
  + infra/k8s/ingress.yaml: sigam.example.com and api.sigam.example.com
* Optional: storageClassName for PVCs depending on your cluster.

Build and deploy (summary)

* Build and push images (from infra/README.md):
  + docker build -t YOUR\_REGISTRY/sigam/server:latest -f server/Dockerfile .
  + docker build --build-arg VITE\_API\_URL=[https://api.sigam.example.com](https://api.sigam.example.com/) -t YOUR\_REGISTRY/sigam/client:latest -f client/Dockerfile .
  + docker push both images
* Apply K8s manifests:
  + kubectl apply -f infra/k8s/namespace.yaml
  + kubectl apply -f infra/k8s/db-secret.yaml
  + kubectl apply -f infra/k8s/postgres-statefulset.yaml
  + kubectl apply -f infra/k8s/postgres-backup-pvc.yaml
  + kubectl apply -f infra/k8s/postgres-backup-cronjob.yaml
  + kubectl apply -f infra/k8s/server-deployment.yaml
  + kubectl apply -f infra/k8s/client-deployment.yaml
  + kubectl apply -f infra/k8s/ingress.yaml

Notes and assumptions

* TypeORM config: I made a minimal change to use env variables (PGHOST, PGPORT, PGUSER, PGPASSWORD, PGDATABASE). Prisma already uses DATABASE\_URL.
* Static files path: The server image runs with WORKDIR /app so process.cwd() becomes /app and your existing static path join(process.cwd(), 'server', 'public') resolves correctly to /app/server/public.
* Client envs: The Vite client reads the API URL at build time. Build a client image per environment (staging/prod) with the correct --build-arg VITE\_API\_URL.
* Ingress controller: The manifests assume NGINX Ingress is installed. Adjust annotations/TLS to your environment as needed.
* Storage: PVCs use the default storage class. If your cluster needs a specific one, set storageClassName.