React + TypeScript + Vite UI on AKS with GitLab CI/CD

Complete handbook containing all project code, Kubernetes manifests, and documentation.

# 1. Introduction & Features

This handbook documents the React + TypeScript + Vite UI application deployed on Azure Kubernetes Service (AKS) with GitLab CI/CD.  
It includes application code, deployment configurations, Kubernetes manifests, and supporting documentation.

# 2. Folder Structure

react-ui-vite-app/  
├── .gitlab-ci.yml  
├── Dockerfile  
├── nginx.conf  
├── package.json  
├── tsconfig.json  
├── vite.config.ts  
├── index.html  
├── src/  
│ ├── App.tsx  
│ └── main.tsx  
├── .env.development  
├── .env.uat  
├── .env.production  
├── k8s/  
│ ├── base/  
│ │ ├── deployment.yaml  
│ │ ├── service.yaml  
│ │ └── ingress.yaml  
│ ├── dev/kustomization.yaml  
│ ├── uat/kustomization.yaml  
│ ├── prod/kustomization.yaml  
│ ├── namespaces/  
│ │ ├── namespace-dev.yaml  
│ │ ├── namespace-uat.yaml  
│ │ └── namespace-prod.yaml  
│ └── rbac/  
│ ├── gitlab-deployer-dev.yaml  
│ ├── gitlab-deployer-uat.yaml  
│ └── gitlab-deployer-prod.yaml  
├── docs/  
│ ├── README.md  
│ ├── MVP\_SETUP.md  
│ ├── KUBECTL-COMMANDS.md  
│ ├── TROUBLESHOOTING.md  
│ └── ROADMAP.md

# 3. Application Code

# 4. Deployment Code

# 5. Kubernetes Manifests

# 6. Documentation

## docs/README.md

# React + TypeScript + Vite UI on AKS via GitLab CI/CD  
  
Production-ready skeleton for a \*\*React + TypeScript + Vite\*\* frontend deployed to \*\*Azure Kubernetes Service (AKS)\*\* with \*\*GitLab CI/CD\*\* across \*\*Dev / UAT / Prod\*\* namespaces.  
  
---  
  
## 🔹 Features  
- Dev/UAT/Prod deployments via GitLab CI/CD   
 - Dev & UAT → auto after MR merge   
 - Prod → requires manual approval   
- Environment configs via `.env.development`, `.env.uat`, `.env.production` (`VITE\_` prefix)   
- Kustomize overlays per environment (namespace, host, image tag)   
- Dockerized static site served by Nginx   
- Optional scoped RBAC for GitLab deployer   
- Clear roadmap for Phase 2+ (TLS, ConfigMaps, Helm, monitoring)  
  
---  
  
## 🔹 Prerequisites  
Ensure the following tools are installed locally (for testing/building):  
- [Node.js](https://nodejs.org/) (>=18.x) & npm  
- [Docker](https://docs.docker.com/get-docker/)  
- [kubectl](https://kubernetes.io/docs/tasks/tools/)  
- [kustomize](https://kubectl.docs.kubernetes.io/installation/kustomize/)  
- Access to an \*\*Azure AKS cluster\*\* & \*\*Azure Container Registry (ACR)\*\*  
- A GitLab repository  
  
---  
  
## 🔹 CI/CD Pipeline Flow  
  
```mermaid  
graph LR  
 A[Feature Branch] -->|Commit| B[Build Only]  
 B -->|Merge to dev| C[Deploy to Dev]  
 C -->|Merge to uat| D[Deploy to UAT]  
 D -->|Merge to main| E[Manual Approval → Deploy to Prod]  
```  
  
---  
  
## 🔹 Local Development  
  
Run the app locally using Vite dev server:  
  
```bash  
npm install  
npm run dev # start dev server at http://localhost:5173  
npm run build # build production bundle to /dist  
npm run preview # preview production build locally  
```  
  
Or build & run inside Docker:  
  
```bash  
docker build --build-arg ENV=development -t react-ui:local .  
docker run -p 8080:80 react-ui:local  
```  
  
---  
  
## 🔹 Environment Variables (Vite)  
  
Environment-specific files at repo root:  
```  
.env.development  
.env.uat  
.env.production  
```  
  
All variables \*\*must be prefixed with `VITE\_`\*\* to be accessible in React.  
  
Example `.env.uat`:  
```env  
VITE\_API\_URL=https://uat-api.mycompany.com  
VITE\_ENV=uat  
```  
  
---  
  
## 🔹 Kubernetes Setup  
  
1. Create namespaces:  
```bash  
kubectl apply -f k8s/namespaces/  
```  
  
2. Deploy per environment (done automatically in CI/CD):  
```bash  
kubectl apply -k k8s/dev  
kubectl apply -k k8s/uat  
kubectl apply -k k8s/prod  
```  
  
---  
  
## 🔹 DNS / Ingress Notes  
Update DNS to point to your AKS ingress controller IP:  
- `reactui-dev.mycompany.com` → Dev namespace  
- `reactui-uat.mycompany.com` → UAT namespace  
- `reactui.mycompany.com` → Prod namespace  
  
---  
  
## 🔹 GitLab CI/CD  
  
The pipeline (`.gitlab-ci.yml`) includes:  
- \*\*Build stage\*\* → builds Docker image with correct `.env` file, pushes to ACR  
- \*\*Deploy stages\*\*:  
 - `dev` branch → auto deploy to Dev namespace  
 - `uat` branch → auto deploy to UAT namespace  
 - `main` branch → manual approval required → deploy to Prod namespace  
  
---  
  
## 🔹 Optional: GitLab Secrets & RBAC  
  
- For \*\*MVP\*\*, the pipeline uses a single `KUBE\_TOKEN` (cluster-admin).   
- Later (Phase 2), enable scoped ServiceAccounts (`k8s/rbac/`) and add GitLab secrets:  
 - `KUBE\_TOKEN\_DEV`  
 - `KUBE\_TOKEN\_UAT`  
 - `KUBE\_TOKEN\_PROD`  
- Replace the token in `.gitlab-ci.yml` with these values.  
  
---  
  
## 🔹 Troubleshooting  
  
- \*\*Pod stuck in ImagePullBackOff\*\*   
 → Check ACR login credentials in GitLab CI/CD variables.   
  
- \*\*Ingress not working\*\*   
 → Ensure DNS points to AKS ingress controller.   
 → Check `kubectl get ingress -n <namespace>`.   
  
- \*\*Environment variables not loading\*\*   
 → Ensure they start with `VITE\_`.   
 → Rebuild the app (`docker build ...`) after changes.   
  
---  
  
## 🔹 Future Improvements (Phase 2+)  
  
- Add \*\*TLS/HTTPS\*\* via cert-manager for Ingress   
- Use \*\*Git commit SHA\*\* for image tags (rollback support)   
- Add lint/tests + image scanning in CI pipeline   
- Use \*\*ConfigMaps/Secrets\*\* for runtime configs (instead of baking envs into Docker image)   
- Add \*\*monitoring & logging\*\* (Prometheus, Grafana, Loki, Azure Monitor)   
- Consider \*\*Helm\*\* for easier versioning & rollbacks when app grows   
  
---  
  
✅ With this README, your team has everything:   
- Setup guide   
- CI/CD flow (with diagram)   
- Local dev instructions   
- DNS/Ingress notes   
- Troubleshooting tips   
- Roadmap for Phase 2+

## docs/MVP\_SETUP.md

# ✅ MVP Setup Checklist  
  
Follow these steps to get the React + TS + Vite app deployed via GitLab CI/CD to AKS.  
  
---  
  
## 1. Prerequisites  
- Azure AKS cluster running  
- Azure Container Registry (ACR)  
- GitLab project created  
  
---  
  
## 2. Configure GitLab Variables  
Go to \*\*Settings > CI/CD > Variables\*\* in GitLab and add:  
  
- `AZURE\_ACR\_USERNAME`  
- `AZURE\_ACR\_PASSWORD`  
- `KUBE\_SERVER`  
- `KUBE\_CA`  
- `KUBE\_TOKEN` (🔸 cluster-admin for MVP; later replace with RBAC tokens)  
  
---  
  
## 3. Kubernetes Setup  
Apply namespaces:  
```bash  
kubectl apply -f k8s/namespaces/  
```  
  
Verify:  
```bash  
kubectl get ns  
```  
  
---  
  
## 4. DNS & Ingress  
- Update DNS records for:  
 - `reactui-dev.mycompany.com`  
 - `reactui-uat.mycompany.com`  
 - `reactui.mycompany.com`  
- Ensure they point to your AKS ingress controller.  
  
---  
  
## 5. Pipeline Flow  
- Create feature branch → run builds only.  
- Merge to `dev` → auto deploy to Dev namespace.  
- Merge to `uat` → auto deploy to UAT namespace.  
- Merge to `main` → manual approval required → deploy to Prod namespace.  
  
---  
  
## 6. Branch Protection (GitLab)  
- Protect `dev`, `uat`, `main` branches.  
- Allow merges only via Merge Requests.  
- Require reviews as per environment:  
 - Dev → 1 reviewer  
 - UAT → 2 reviewers  
 - Prod → 2+ reviewers  
  
---  
  
✅ With this, you have a fully working MVP setup.   
Later, see README Phase 2 for hardening (TLS, RBAC, monitoring).

## docs/KUBECTL-COMMANDS.md

# 📌 Useful kubectl Commands for AKS (uk8s)  
  
This cheat sheet provides the most common `kubectl` commands with \*\*comments\*\* for working with \*\*Azure Kubernetes Service (AKS / uk8s)\*\* in Dev, UAT, and Prod environments.  
  
---  
  
## 🔹 Context & Cluster Info  
```bash  
kubectl config get-contexts # List all contexts  
kubectl config use-context <context> # Switch to AKS context  
kubectl cluster-info # Show AKS cluster info (API server, DNS, etc.)  
kubectl get nodes -o wide # List AKS nodes with IPs  
kubectl describe node <node-name> # Detailed node info  
```  
  
---  
  
## 🔹 Namespaces  
```bash  
kubectl get ns # List all namespaces  
kubectl create ns at41457-dev-recsdiui-dev # Create Dev namespace  
kubectl delete ns at41457-uat-recsdiui-uat # Delete UAT namespace  
kubectl config set-context --current --namespace=at41457-dev-recsdiui-dev # Set default namespace  
```  
  
---  
  
## 🔹 Deployments & Pods  
```bash  
kubectl get deployments -n at41457-dev-recsdiui-dev # List deployments in Dev  
kubectl describe deployment react-ui -n at41457-uat-recsdiui-uat # Debug deployment in UAT  
kubectl rollout status deployment react-ui -n at41457-prod-recsdiui-prod # Check rollout progress in Prod  
kubectl rollout undo deployment react-ui -n at41457-prod-recsdiui-prod # Rollback last Prod deployment  
  
kubectl get pods -n at41457-dev-recsdiui-dev # List pods in Dev  
kubectl describe pod <pod-name> -n at41457-dev-recsdiui-dev # Debug pod  
kubectl logs <pod-name> -n at41457-dev-recsdiui-dev # View pod logs  
kubectl logs -f <pod-name> -n at41457-dev-recsdiui-dev # Stream logs  
kubectl exec -it <pod-name> -n at41457-uat-recsdiui-uat -- sh # Exec into pod  
```  
  
---  
  
## 🔹 Services & Ingress  
```bash  
kubectl get svc -n at41457-dev-recsdiui-dev # List services in Dev  
kubectl describe svc react-ui-service -n at41457-uat-recsdiui-uat # Debug service in UAT  
  
kubectl get ingress -n at41457-prod-recsdiui-prod # List ingress rules in Prod  
kubectl describe ingress react-ui-ingress -n at41457-prod-recsdiui-prod # Debug ingress  
```  
  
---  
  
## 🔹 ConfigMaps & Secrets  
```bash  
kubectl get configmaps -n at41457-dev-recsdiui-dev # List configmaps in Dev  
kubectl describe configmap <name> -n at41457-uat-recsdiui-uat # Debug a configmap  
  
kubectl get secrets -n at41457-prod-recsdiui-prod # List secrets in Prod  
kubectl describe secret <name> -n at41457-prod-recsdiui-prod # Debug a secret  
kubectl get secret <name> -n at41457-prod-recsdiui-prod -o jsonpath="{.data.key}" | base64 --decode # Decode a secret  
```  
  
---  
  
## 🔹 Events & Debugging  
```bash  
kubectl get events -n at41457-dev-recsdiui-dev --sort-by=.metadata.creationTimestamp # Recent events in Dev  
kubectl describe pod <pod-name> -n at41457-uat-recsdiui-uat # Pod details  
kubectl top pods -n at41457-prod-recsdiui-prod # Pod CPU/Memory usage  
kubectl top nodes # Node CPU/Memory usage  
```  
  
---  
  
## 🔹 Apply & Delete Manifests  
```bash  
kubectl apply -f k8s/namespaces/namespace-dev.yaml # Apply single manifest  
kubectl apply -k k8s/dev # Apply Kustomize overlay (Dev)  
  
kubectl delete -f k8s/namespaces/namespace-uat.yaml # Delete UAT namespace manifest  
kubectl delete -k k8s/prod # Delete all Prod resources  
```  
  
---  
  
## 🔹 Scaling  
```bash  
kubectl scale deployment react-ui --replicas=3 -n at41457-uat-recsdiui-uat # Scale UAT deployment to 3 pods  
kubectl autoscale deployment react-ui -n at41457-prod-recsdiui-prod --min=2 --max=5 --cpu-percent=70 # HPA  
```  
  
---  
  
## 🔹 Access Application (Port-Forward)  
```bash  
kubectl port-forward svc/react-ui-service 3000:80 -n at41457-dev-recsdiui-dev  
# Opens http://localhost:3000 → maps to Dev service port 80  
```  
  
---  
  
✅ These commands cover \*\*90% of daily AKS operations\*\*:   
- Managing \*\*namespaces, pods, deployments\*\*   
- Debugging issues (logs, events, describe)   
- Checking \*\*Ingress/DNS\*\*   
- Scaling, rolling back, port-forward for local testing

## docs/TROUBLESHOOTING.md

# 🚑 Troubleshooting Guide  
  
This document will track \*\*common issues\*\* and their \*\*solutions\*\* when working with the React UI app on AKS.  
  
---  
  
## 🔹 Common Issues  
  
- \*\*Pod stuck in ImagePullBackOff\*\*  
 - Check if Docker image exists in ACR and GitLab CI/CD variables for ACR login are correct.  
  
- \*\*Ingress not resolving\*\*  
 - Ensure DNS records are configured and pointing to AKS ingress controller IP.  
  
- \*\*Environment variables missing\*\*  
 - Vite requires `VITE\_` prefix for all env vars. Double-check `.env.\*` files.  
  
---  
  
## 🔹 TODO  
- Expand with real-world issues as they occur.

## docs/ROADMAP.md

# 📍 Roadmap for React UI on AKS  
  
## ✅ Phase 1 – Working Setup (MVP)  
- React + TS + Vite starter app  
- GitLab CI/CD for Dev, UAT, Prod  
- Namespaces + Kustomize overlays  
- Optional RBAC (Dev example)  
  
## 🔒 Phase 2 – Production Readiness  
- TLS/HTTPS with cert-manager  
- Git commit SHA tagging for rollbacks  
- Scoped RBAC for UAT/Prod  
- Branch protections & reviews  
  
## 📈 Phase 3 – Enterprise Grade  
- Monitoring (Prometheus, Grafana)  
- Centralized logs (Loki/ELK/Azure Monitor)  
- ConfigMaps/Secrets for runtime configs  
- CDN + WAF (Azure Front Door/Cloudflare)  
- Helm adoption for templated rollouts  
  
---  
This roadmap ensures the project grows from MVP → Production → Enterprise-ready.