INSURANCE MANAGEMENT SYSTEM SETUP GUIDE
==========

1. Prerequisites / Dependencies

Before running the project, make sure you have the following installed:

- Git
- Docker
- Docker Compose
- Kubernetes (kubectl)
- Python 3.x
- Node.js & npm (for Angular frontend)
- Optional: Helm (if you plan to manage Kubernetes charts)

Python dependencies for backend services:
- Flask
- Flask-CORS
- SQLAlchemy
- Werkzeug
- Other dependencies listed in `requirements.txt` files for each service

Angular frontend dependencies:
- Install node modules using:
  npm install
  (run this inside the `frontend/` directory)

---

2. Docker Commands

To build and run all services locally using Docker Compose:

```
# Build Docker images for all services
docker-compose build

# Start containers in detached mode
docker-compose up -d

# Stop containers
docker-compose down

# Optional: clean up dangling images
docker system prune -f
```

---

3. Kubernetes Commands

To deploy services to a Kubernetes cluster:

```
# Check current kubectl context
kubectl config current-context

# Get cluster info
kubectl cluster-info

# Apply all Kubernetes manifests (Deployments, Services, Ingress)
kubectl apply -f k8s/
```

```
# Check all pods
kubectl get pods

# Check services
kubectl get svc

# Optional: Delete all resources
kubectl delete -f k8s/
```

---

4. Notes

- Backend services: `auth-service`, `claim-service`, `policy-service`
- API Gateway routes requests to the backend services
- Frontend is served by Angular + Nginx
- Default admin user for auth-service:
  username: admin
  password: admin123
- Make sure ports in docker-compose and Kubernetes manifests do not conflict