






# Lab Report: Modern To-Do App with Next.js, Supabase, and Kubernetes

## 1. Introduction

This report documents the development and deployment of a **Modern To-Do List Application** built with:

- **Next.js** (React framework) for the frontend.
- **Supabase** (PostgreSQL) for real-time database and authentication.
- **shadcn/ui** for beautiful, accessible UI components.
- **Docker** for containerization.
- **Kubernetes** (on Google Cloud VM) for scalable deployment.

The app features:

-  **Task management** (Create, Read, Update, Delete)
-  **Due dates and priority levels**
-  **Real-time updates** (via Supabase)
-  **User authentication** (Supabase Auth)
-  **Scalable cloud deployment**

## 2. Technology Stack

Component	Technology Used
Frontend	Next.js, React, TypeScript
UI Library	shadcn/ui, Tailwind CSS
Database	Supabase (PostgreSQL)
Authentication	Supabase Auth
Containerization	Docker
Orchestration	Kubernetes (GKE)
Cloud Platform	Google Cloud Platform (GCP)

## 3. Key Code Snippets

### A. Next.js API Route (Task Creation)

All the APIs for tasks are done using Supabase. These codes can be found in the `actions/todos/actions.tsx` file. Following is code for one of the actions i.e. add new to-do task.

```
// actions/todos/actions.tsx
import { createClient } from '@supabase/supabase-js';
```

```
export async function addTodo(formData: FormData) {
  const supabase = createClient();

  const {
    data: { user },
  } = await supabase.auth.getUser();

  const { error } = await supabase
    .from("todos")
    .insert([
      {
        user_id: user?.id,
        task: formData.get("task") as string,
        is_complete: false,
        inserted_at: new Date(),
        priority: (formData.get("priority") as string) || "medium",
        due_date: (formData.get("due_date") as string) || null,
      },
    ])
    .select();

  if (error) {
    throw new Error(error.message);
  }

  revalidatePath("/");
}
```

## B. Task List Component

```
// components/todos/todos.tsx
import { createClient } from "@utils/supabase/server";
import Todo from "../todo";
import AddTodo from "../add-todo";

export default async function Todos() {
  const supabase = createClient();

  const { data: todos, error } = await supabase
    .from("todos")
    .select("*")
    .order("priority", { ascending: true }) // High priority first
    .order("due_date", { ascending: true }) // Soonest dates first
    .order("inserted_at", { ascending: true });
  if (error) {
    throw new Error(error.message);
  }

  return (
    <div className="flex-1 overflow-auto">
```

```

    <div className="flex flex-col">
      {todos &&
        todos
          .filter((todo) => {
            return todo.is_complete == false;
          })
          .map((todo) => {
            return <Todo key={todo.id} todo={todo} />;
          })
      }
      {todos &&
        todos
          .filter((todo) => {
            return todo.is_complete;
          })
          .map((todo) => {
            return <Todo key={todo.id} todo={todo} />;
          })
      }
      <AddTodo />
    </div>
  </div>
);
}

```

## 4. Deployment Steps

### A. Dockerizing the App

#### 1. Create **Dockerfile**

```

# Stage 1: Build
FROM node:18-alpine AS builder
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

# Stage 2: Run
FROM node:18-alpine
WORKDIR /app
ENV NODE_ENV=production
COPY --from=builder /app/package*.json ./
COPY --from=builder /app/node_modules ./node_modules
COPY --from=builder /app/.next ./next
COPY --from=builder /app/public ./public
EXPOSE 3000
CMD ["npm", "start"]

```

#### 2. Build and Push to Docker Hub

```
docker build -t swoichha/todo-app-main:latest .
docker push swoichha/todo-app-main:latest
```

---

## B. Deploying to Google Cloud VM

### 1. Set Up VM

```
gcloud compute instances create todo-vm \
  --zone=us-central1-a \
  --machine-type=e2-medium \
  --tags=http-server
```

### 2. SSH into VM and Install Tools

```
gcloud compute ssh todo-vm --zone=us-central1-a

# Install Docker & Kubernetes
sudo apt update && sudo apt install -y docker.io kubectl
sudo systemctl enable docker
sudo systemctl start docker
```

### 3. Deploy with Kubernetes

#### deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: todo-app-main
spec:
  replicas: 2
  selector:
    matchLabels:
      app: todo-app-main
  template:
    metadata:
      labels:
        app: todo-app-main
    spec:
      containers:
        - name: todo-app-main
          image: swoichha/todo-app-main:latest
          ports:
            - containerPort: 3000
```

```
env:
  - name: NEXT_PUBLIC_SUPABASE_URL
    value: "https://sjrwjctowjtvasrnyvt.supabase.co"
  - name: NEXT_PUBLIC_SUPABASE_ANON_KEY
    value:
      "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSI6InJlZiI6InNq
      cndqY3Rvd2p0dmFzcm52eXZ0Iiwicm9sZSI6ImFub24iLCJpYXQiOiJE3NDU0NDgyMjQsImV4cC
      I6MjA2MTAyNDIyNH0.LbgN1hPTi1Y6gF7g4Rqe97mremDgM_QErXsI0K-_PtY"
```

### service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: todo-app-main-service
spec:
  selector:
    app: todo-app-main
  ports:
    - protocol: TCP
      port: 80
      targetPort: 3000
  type: LoadBalancer
```

### Apply Configs:

```
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
```

---

## 5. Accessing the App

### 1. Get the External IP:

```
kubectl get service todo-app-main-service
```

Example Output:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
PORT(S)			
todo-app-main-service	LoadBalancer	10.0.123.45	203.0.113.45
80:3000/TCP			

### 2. Open in Browser:

http://203.0.113.45

## 6. Key Features Implemented

Feature	How It Works
Real-time Sync	Supabase PostgreSQL listens for changes
Priority Tags	high/medium/low badges with color coding
Due Dates	Formatted with date-fns
Auth	Supabase Magic Links or Email/Password
Responsive UI	Tailwind CSS + shadcn/ui components

There are many other features that have implemented like:

- Add: Write the task and click the Add Task button
- Edit: On an exsiting task just click on the title and start editing it
- Completed task: Using a checkbox user can mark the task as completed
- Delete: User can delete any task by marking a task is checked as complete and clicking on Clear Completed button or can delete all the task in the list by clicking on the Clear All button
- Priority: While creating a task user can assign the priory for each task and the list of task will be sorted based on the priority level followed by the deadling in case two or more task have same level of priority.

## 7. Troubleshooting

Issue	Solution
Pods not starting	kubectl logs <pod-name>
No external IP	Wait 2 mins or check firewall rules
Supabase connection failed	Verify NEXT_PUBLIC_SUPABASE_URL

## 8. Conclusion

We successfully built and deployed a **scalable To-Do app** using:

- **Next.js** for a fast frontend.
- **Supabase** for real-time data and auth.
- **Kubernetes** for automated scaling.
- **Google Cloud** for production hosting.

### Future Improvements:

- Implement push notifications for due dates.

- Set up CI/CD with GitHub Actions.