EC2 frontend changes not applied

01/04/2026 15:12:57

i have made changes in code then push to the github after that i pull tho...

Step 1: Verify the pull Run on your EC2: bash Copy code `git...

the issue i am facing is that when i first visit website it automatically...

GNU nano 7.2 page.tsx 'use client'; import { useEffect } from 'react';...

task-management/frontend/src/app/page.tsx i am taking about this...

"use client"; import { useState } from "react"; import Cookies from "js-...

1. **Make `auth_token` persistent and accessible across all routes:** ts...

is there need in with config of nginx?

still same issue i have tried everything

In `ManagerDashboard`: ❌ Remove: ts Copy code `const router =...

what is vpc peering

elastic cache for redis ?

i have made changes in code then push to the github after that i pull those changes on ec2 on aws by command git pull origin main and then create build and started the frontend but still those changes are not applied

Step 1: Verify the pull Run on your EC2: bash Copy code `git status git log -1` * `git status` should show nothing to commit (clean). * `git log -1` should show the latest commit that you pushed. If the latest commit is **not there**, the pull didn’t actually update your…

```
ubuntu@ip-10-0-1-52:~/task-management/frontend$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   package-lock.json

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ecosystem.config.js

no changes added to commit (use "git add" and/or "git commit -a")
ubuntu@ip-10-0-1-52:~/task-management/frontend$ git log -1
commit cb99181a47cf75f720bf20cd835a3e693bb8e47b (HEAD -> main, origin/main, origin/HEAD)
Author: Abdullah Qureshi <abdullahqureshionline@gmail.com>
Date:   Thu Dec 25 23:14:39 2025 +0500

    routing corrected
```

the issue i am facing is that when i first visit website it automatically shows me login page and i add credential it takes me to the dashboard then i close the website after closing website i again visit the website it shows me blank page when i add dashboard to the url then it shows me dashboard else not how to solve this issue

```
  GNU nano 7.2                    page.tsx
'use client';

import { useEffect } from 'react';
import { useRouter } from 'next/navigation'; // ← Correct
import for App Ro>
import Cookies from 'js-cookie';

export default function Home() {
  const router = useRouter();

  useEffect(() => {
    const token = Cookies.get('auth_token');

    if (token) {
      router.replace('/dashboard'); // Logged-in →
dashboard
    } else {
      router.replace('/login'); // Not logged in → login
    }
  }, [router]);

  return null; // Nothing rendered while redirecting
}
```

on page.tsx i have this code

task-management/frontend/src/app/page.tsx
i am taking about this page.tsx where this code is pasted

```
"use client";

import { useState } from "react";
import Cookies from "js-cookie";
import { useRouter } from "next/navigation";
export default function Login() {
  const router = useRouter();
  const [formData, setFormData] = useState({
    email: "",
    password: "",
  });

  const handleChange = (
    e: React.ChangeEvent<HTMLInputElement |
HTMLSelectElement>
  ) => {
    const { name, value } = e.target;
    setFormData({ ...formData, [name]: value });
  };

  const handleSubmit = async (e:
React.FormEvent<HTMLFormElement>) => {
    e.preventDefault();
    try {
      const response = await fetch("/api/user/login", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(formData),
      });
      if (!response.ok) {
        throw new Error("Login Failed");
      }
      const data = await response.json();
      Cookies.set("auth_token", data.Data.token, {
        expires: 1,
        sameSite: "lax",
      });

      localStorage.setItem("UserName",
```

```
data.Data.username);
      localStorage.setItem("Email", data.Data.email);
      router.push("/dashboard");
    } catch (error) {
      console.log(error);
    }
  };
"use client";

import { useState } from "react";
import Cookies from "js-cookie";
import { useRouter } from "next/navigation";
export default function Signup() {
  const router = useRouter();
  const [formData, setFormData] = useState({
    username: "",
    email: "",
    password: "",
    role: "",
  });

  const handleChange = (
    e: React.ChangeEvent<HTMLInputElement |
HTMLSelectElement>
  ) => {
    const { name, value } = e.target;
    setFormData({ ...formData, [name]: value });
  };

  const handleSubmit = async (e:
React.FormEvent<HTMLFormElement>) => {
    e.preventDefault();
    try {
      const response = await fetch("/api/user/signup", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(formData),
      });
      if (!response.ok) {
        throw new Error("Signup Failed");
      }
```

```
      const data = await response.json();
      console.log("Response: " , data)
      localStorage.setItem("UserName",
data.data.username);
      localStorage.setItem("Email", data.data.email);
      Cookies.set("auth_token", data.data.token, {
        expires: 1,
        sameSite: "lax",
      });
      router.push("/dashboard");
    } catch (error) {
      console.log(error);
    }
  };
import { useState } from "react";

export default function AddTaskForm({
  token,
  onTaskAdded,
  onClose,
}: {
  token?: string;
  onTaskAdded: (newTask: any) => void;
  onClose: () => void;  // optional
}) {
  const [formData, setFormData] = useState({
    title: "",
    description: "",
    assignedTo: "",
    startDate: "",
    endDate: "",
  });

  if (!token) {
    console.error("No token found!");
    return <p>No token provided. Please log in.</p>;
  }

  const handleChange = (
    e: React.ChangeEvent<
      HTMLInputElement | HTMLSelectElement |
```

```
HTMLTextAreaElement
    >
  ) => {
    const { name, value } = e.target;
    setFormData({ ...formData, [name]: value });
  };

  const handleSubmit = async (e:
React.FormEvent<HTMLFormElement>) => {
    e.preventDefault();

    const payload = {
      ...formData,
      assignedTo: formData.assignedTo ?
[formData.assignedTo] : [],
      startDate: new Date(formData.startDate),
      endDate: new Date(formData.endDate),
    };

    try {
      const response = await fetch("/api/task", {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
          Authorization: Bearer ${token},
        },
        body: JSON.stringify(payload),
      });
      if (!response.ok) {
        throw new Error("Failed to add task");
      }
       const data = await response.json();
      onTaskAdded(data.Task)
      console.log("Task Added Successfully");
      onClose();
    } catch (error) {
      console.error(error);
    }
  };
import { useEffect, useState } from "react";
```

```
export default function EditTask({
  token,
  taskId,
  onClose,
  onTaskUpdate,
}: {
  token?: string;
  taskId: string;
  onClose: () => void;
  onTaskUpdate: (newTask: any) => void;
}) {
  if (!token) {
    console.error("No token found!");
    return <p>No token provided. Please log in.</p>;
  }

  const [formData, setFormData] = useState({
    title: "",
    description: "",
    startDate: "",
    endDate: "",
  });

  const [assignedUsers, setAssignedUsers] =
useState<string[]>([]);
  const [newUserEmail, setNewUserEmail] =
useState("");

  // Fetch task details
  useEffect(() => {
    const fetchTask = async () => {
      try {
        const response = await fetch(
          `/api/task/id/${taskId}`,
          {
            method: "GET",
            headers: { Authorization: `Bearer ${token}` },
          }
        );
        if (!response.ok) throw new Error("Failed to fetch
task");
```

```
        const data = await response.json();
        const task = data.Task;

        const formatDate = (dateString: string) => {
          if (!dateString) return "";
          const date = new Date(dateString);
          return date.toISOString().split("T")[0];
        };

        setFormData({
          title: task.Title || "",
          description: task.Description || "",
          startDate: formatDate(task.StartDate),
          endDate: formatDate(task.EndDate),
        });

        setAssignedUsers(task.AssignedTo || []);
      } catch (error) {
        console.error(error);
      }
    };
    fetchTask();
  }, [taskId, token]);

  // Handle text inputs
  const handleChange = (
    e: React.ChangeEvent<HTMLInputElement |
HTMLTextAreaElement>
  ) => {
    const { name, value } = e.target;
    setFormData((prev) => ({ ...prev, [name]: value }));
  };


  // Remove user
  const handleRemoveUser = (email: string) => {
    setAssignedUsers((prev) => prev.filter((u) => u !==
email));
  };
```

```
  // Add new user
  const handleAddUser = () => {
    if (newUserEmail &&
!assignedUsers.includes(newUserEmail)) {
      setAssignedUsers((prev) => [...prev,
newUserEmail]);
      setNewUserEmail("");
    }
  };

  // Submit update
  const handleUpdate = async (e: React.FormEvent) =>
{
    e.preventDefault();

  const body: any = {
    title: formData.title,
    description: formData.description,
    startDate: new
Date(formData.startDate).toISOString(),
    endDate: new Date(formData.endDate).toISOString(),
    assignedTo: assignedUsers,
  };




    try {
      const response = await
fetch(`/api/task/${taskId}`, {
        method: "PATCH",
        headers: {
          "Content-Type": "application/json",
          Authorization: `Bearer ${token}`,
        },
        body: JSON.stringify(body),
      });
      const data = await response.json();

       if (!response.ok) {
      throw new Error(data.message || "Failed to update
task");
```

```
    }
      const updatedTask = data.Task;
      onTaskUpdate(updatedTask);
      onClose();
    } catch (error) {
      console.error(error);
    }
  };
"use client";

import React, { useState, useEffect } from "react";
import Cookies from "js-cookie";
import AddTaskForm from "./addTaskModal";
import EditTask from "./editTaskModal";
import { useRouter } from "next/navigation";

export default function ManagerDashboard() {
  const [tasks, setTasks] = useState<any[]>([]);
  const [selectedTaskId, setSelectedTaskId] =
useState<any>(null);
  const [taskDetails, setTaskDetails] = useState<any>
(null);
  const token = Cookies.get("auth_token");
  const [editTaskModal, setEditTaskModal] =
useState(false);
  const [addTaskModal, setAddTaskModal] =
useState(false);
  const [loading, setLoading] = useState(true);
  const [username, setusername] = useState<string |
null>(null);
  const router = useRouter();

  useEffect(() => {
    const fetchTasks = async () => {
      setLoading(true);
      try {
        const response = await fetch("/api/task", {
          method: "GET",
          headers: {
            "Content-Type": "application/json",
            Authorization: Bearer ${token},
```

```
        },
      });

      if (!response.ok) {
        throw new Error("Failed to fetch tasks");
      }

      const data = await response.json();
      setTasks(data.Task);
    } catch (err: any) {
      console.error(err);
    } finally {
      setLoading(false);
    }
  };

  fetchTasks();
}, [token]);

useEffect(() => {
  if (!selectedTaskId) return;

  const fetchTaskById = async () => {
    try {
      const response = await fetch(
        `/api/task/id/${selectedTaskId}`,
        {
          headers: { Authorization: `Bearer ${token}` },
        }
      );
      if (!response.ok) throw new Error("Failed to fetch
task details");

      const data = await response.json();
      setTaskDetails(data.Task);
    } catch (error) {
      console.error(error);
    }
  };

  fetchTaskById();
```

```
  }, [selectedTaskId, token]);

  const handleTaskClick = (id: string) => {
    setSelectedTaskId(id);
  };

  const closeSidebar = () => {
    setSelectedTaskId(null);
    setTaskDetails(null);
  };

  const handleTaskAdded = (newTask: any) => {
    setTasks((prev) => [...prev, newTask]);
  };
  const handleTaskUpdated = async (updatedTask: any)
=> {
    setTasks((prev) =>
      prev.map((task) => (task._id === updatedTask._id ?
updatedTask : task))
    );

    if (selectedTaskId === updatedTask._id) {
      try {
        const response = await fetch(
          /api/task/id/${selectedTaskId},
          {
            headers: { Authorization: Bearer  ${token} },
          }
        );
        if (!response.ok) throw new Error("Failed to fetch
task details");

        const data = await response.json();
        setTaskDetails(data.Task);
      } catch (error) {
        console.error(error);
      }
    }
  };

  const handleLogout = async () => {
```

```
    Cookies.remove("auth_token");
    router.push("/login");
  };
  const handleDeleteTask = async (id: string) => {
    try {
      const response = await fetch(`/api/task/${id}`, {
        method: "DELETE",
        headers: {
          "Content-Type": "application/json",
          Authorization: `Bearer ${token}`,
        },
      });

      if (!response.ok) {
        throw new Error("Failed to delete task");
      }
      setTasks((prevTask) => prevTask.filter((task) =>
task._id !== id));
      console.log("Task deleted successfully");
    } catch (error) {
      console.error("Error Deleting task");
    }
  };

  const completedTasks = tasks.filter((t) => t.status
=== "completed");
  const inProgressTasks = tasks.filter((t) => t.status
=== "in_progress");
  const toDoTasks = tasks.filter((t) => t.status === "to-
do");
  useEffect(() => {
    const data = localStorage.getItem("UserName");
    setusername(data);
  }, [])

  if (loading) {
    return (
      <div className="fixed inset-0 flex items-center
justify-center bg-white bg-opacity-80 z-50">
        <div className="w-16 h-16 border-4 border-
indigo-500 border-t-transparent rounded-full animate-
```

```
spin"></div>
      </div>
    );
  }
this is my total code
```

1. **Make `auth_token` persistent and accessible across all routes:** ts Copy code `Cookies.set("auth_token", token, { expires: 7, path: "/", sameSite: "lax" })`

where which page to update

is there need in with config of nginx?

still same issue
i have tried everything

In `ManagerDashboard`: ❌ Remove: ts Copy code `const router = useRouter();` ❌ Remove: ts Copy code `useEffect(() => { if (!token) { router.replace("/login"); } }, []);` Just assume token exists — middleware guarantees it.

if i donr remove this will it make issue?

what is vpc peering

elastic cache for redis ?