## Package up your project

* cd cards
* zip backend server.js
* mv backend.zip ~
* cd frontend
* zip -r frontend App.tsx main.tsx components pages
* mv frontend.zip ~

You’ve now got two zip files in the user folder of your VM. Get them out onto your main machine somehow (e-mailing them to yourself will work if nothing else does).

## Make the VM

* Log into Lightsail.
* Log into Namecheap.
* Create an instance on Lightsail.
  + Get its IP address.
  + Make an A record for it on Namecheap.
  + Ping your server until the name works.
  + Navigate to http://yourserver and see if it works. (Chrome will complain.)
* Open the terminal for the instance through Lightsail. Then, from the server:
  + Edit .ssh/authorized\_keys
  + Add your ssh key.
* Verify that ssh bitnami@yourserver from your own machine works.
* Get your packaged server over to the VM. From your machine:
  + scp frontend.zip backend.zip bitnami@yourserver:
* Now let’s secure the server. From the server:
  + sudo /opt/bitnami/bncert.tool
  + Enter the working domain name, and provide your e-mail address when prompted.
  + Wait for the cert tool to finish, then try connecting to the server via https.

## Make the app environment

From the server:

* cd /opt/bitnami
* sudo mkdir projects
* sudo chown bitnami projects
* cd projects

## Make the app

Again from the server:

* mkdir cards
* cd cards
* npm create vite@latest frontend
* cd frontend
* npm install

Edit **vite.config.ts**. In the defineConfig block, add the directive:

server: {  
 port: 3000,  
},

## Set up the Apache virtual host

From another window:

* cd /opt/bitnami/apache/conf/vhosts
* cp sample-https-vhost.conf.disabled sample-https-vhost.conf

Edit sample-https-vhost.conf and replace the certificate file paths:

SSLCertificateFile "/opt/bitnami/apache/conf/<yourserver>.crt"  
SSLCertificateKeyFile "/opt/bitnami/apache/conf/<yourserver>.key"

Replace the ProxyPass lines with:

ProxyPass /api/ http://localhost:5000/api/ upgrade=websocket

ProxyPassReverse /api/ http://localhost:5000/api/ upgrade=websocket

ProxyPass / http://localhost:3000/ upgrade=websocket

ProxyPassReverse / http://localhost:3000/ upgrade=websocket

Now restart the server:

* sudo /opt/bitnami/ctlscript.sh restart apache

Back in your cards window:

* npm run dev

Try the server. You should get the Vite example screen.

## Install the dependencies

From the server:

* cd /opt/bitnami/projects/cards
* npm install express --save
* npm install body-parser
* npm install mongodb
* npm install cors
* npm install nodemon
* cd frontend
* npm install react-router-dom

## Unpack the project

Again from the server:

* cd /opt/bitnami/projects/cards
* unzip ~/backend.zip
* cd frontend/src
* unzip ~/frontend.zip

Accept all overwrites.

## Configure the App

Edit package.json in the root directory, and again add our server definition:

{

  "name": "backend",

  "version": "1.0.0",

  "description": "",

  "main": "server.js",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1",

    "start": "nodemon server.js"

  },

  "author": "",

  "license": "ISC",

  "dependencies": {

    "body-parser": "^1.19.0",

    "express": "^4.17.1"

  }

}

## Try it again – sort of

You can do an npm run dev from cards/frontend, connect and verify that you get the login screen, but nothing else will work yet. For the backend, we need MongoDB.

## Create a database

* Log in to Mongo Atlas.
* Create a new project and a new M0 cluster.
* Set the username as cop4331 and enter or get a password. Make sure to copy the password.
* Create the cluster, wait for it to provision, and copy the connection string.
  + Once you get the connection string, make sure you can ping the server.
* Go to Network Access, add the Lightsail IP address, and delete all the others.
* Create a new database “cop4331”, and collections “Users” and “Cards”.
* Add a user to Users.
  + Give it a Login, Password, UserId (this is a number!), FirstName and LastName.

## Link up the database

* Make sure you can ping the cluster server (pull it out of the connection string). It can take a few minutes for it to propagate even when Mongo Atlas says it’s done.
* Look at your connection string. Make sure the correct password is in it, and make sure you add the database name to the end of the path portion of the URL (before ?retryWrites).

## Fix the connections

* Edit cards/server.js and replace the old Mongo URL with your connection string.
* Edit cards/frontend/src/components/CardUI.tsx and Login.tsx, and fix the API calls. **You should be able to figure out how to do this by now. Can you?**

## Start the servers

* Start the servers, with the usual npm start and npm run dev commands.
* Connect to the server – and it should work.