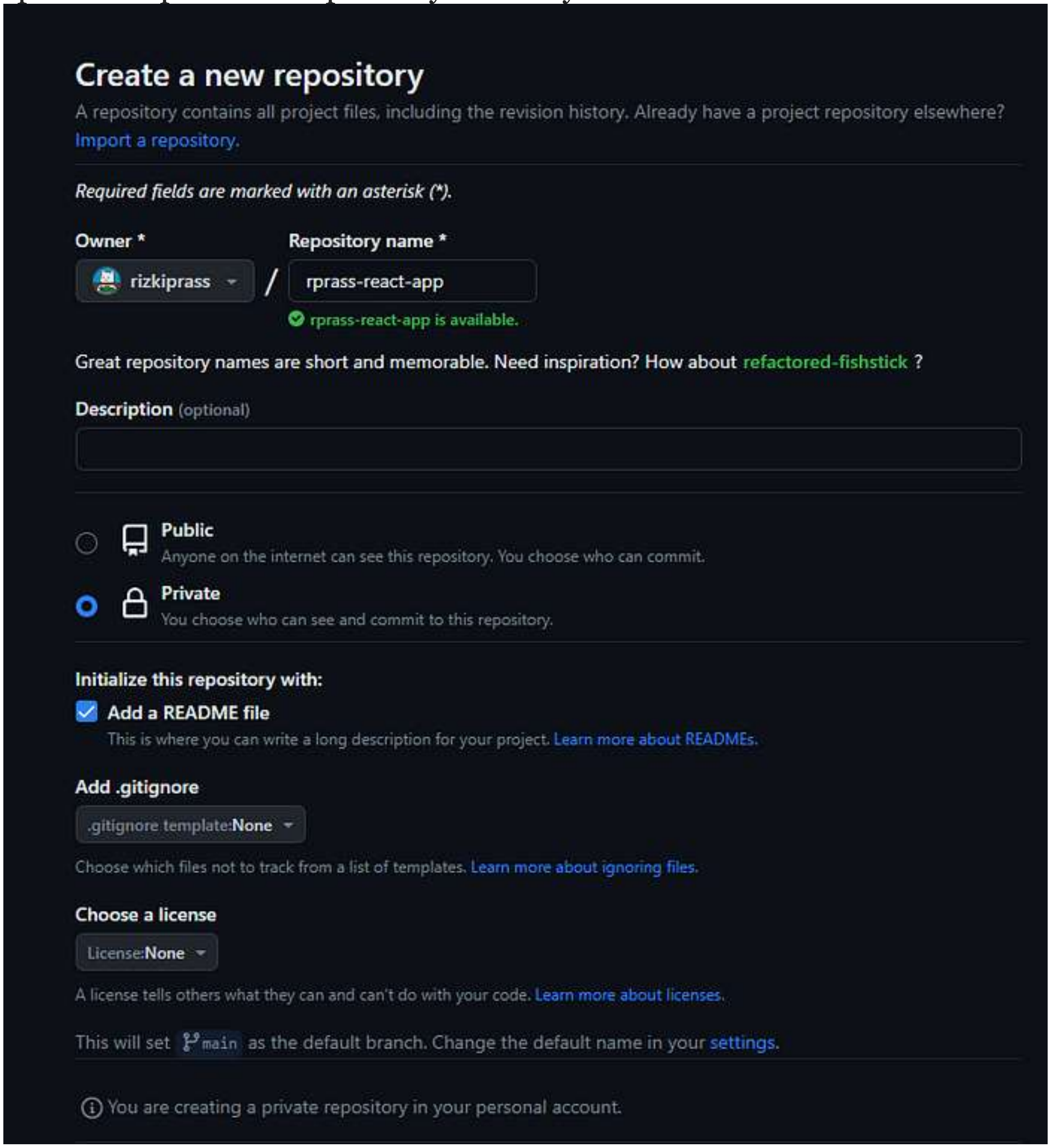
1. Create a GIT repository and setup our React App

Let’s start by creating a new repository, which we will use for our React app. Choose a repository name that you prefer. Then, you can select the “private” option for repository visibility.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner * rizkiprass / Repository name * rpress-react-app

rprass-react-app is available.

Great repository names are short and memorable. Need inspiration? How about refactored-fishstick ?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore
.gitignore template: None
Choose which files not to track from a list of templates. Learn more about ignoring files.

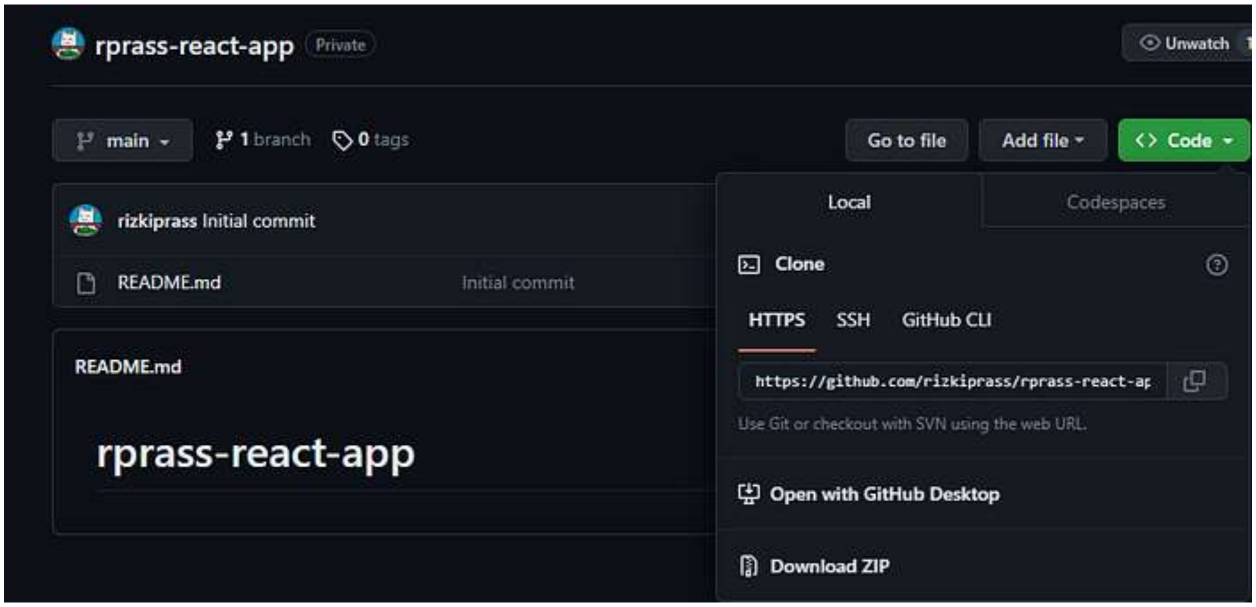
Choose a license
License: None
A license tells others what they can and can't do with your code. Learn more about licenses.

This will set main as the default branch. Change the default name in your settings.

You are creating a private repository in your personal account.

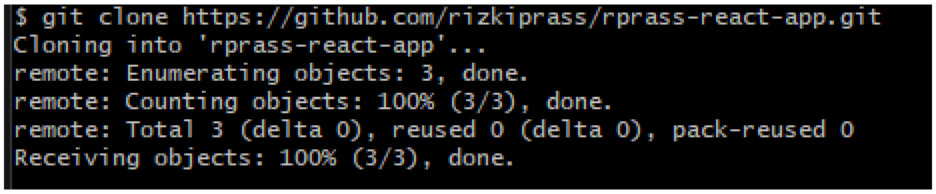
After you finish, click on the “Create Repository” button.

Once the repository is created, proceed to copy the repository URL from the HTTPS tab.



Paste it onto your machine using the following command:

```
git clone <your-repo-url>
```



```
$ git clone https://github.com/rizkipras/rpress-react-app.git
Cloning into 'rpress-react-app'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

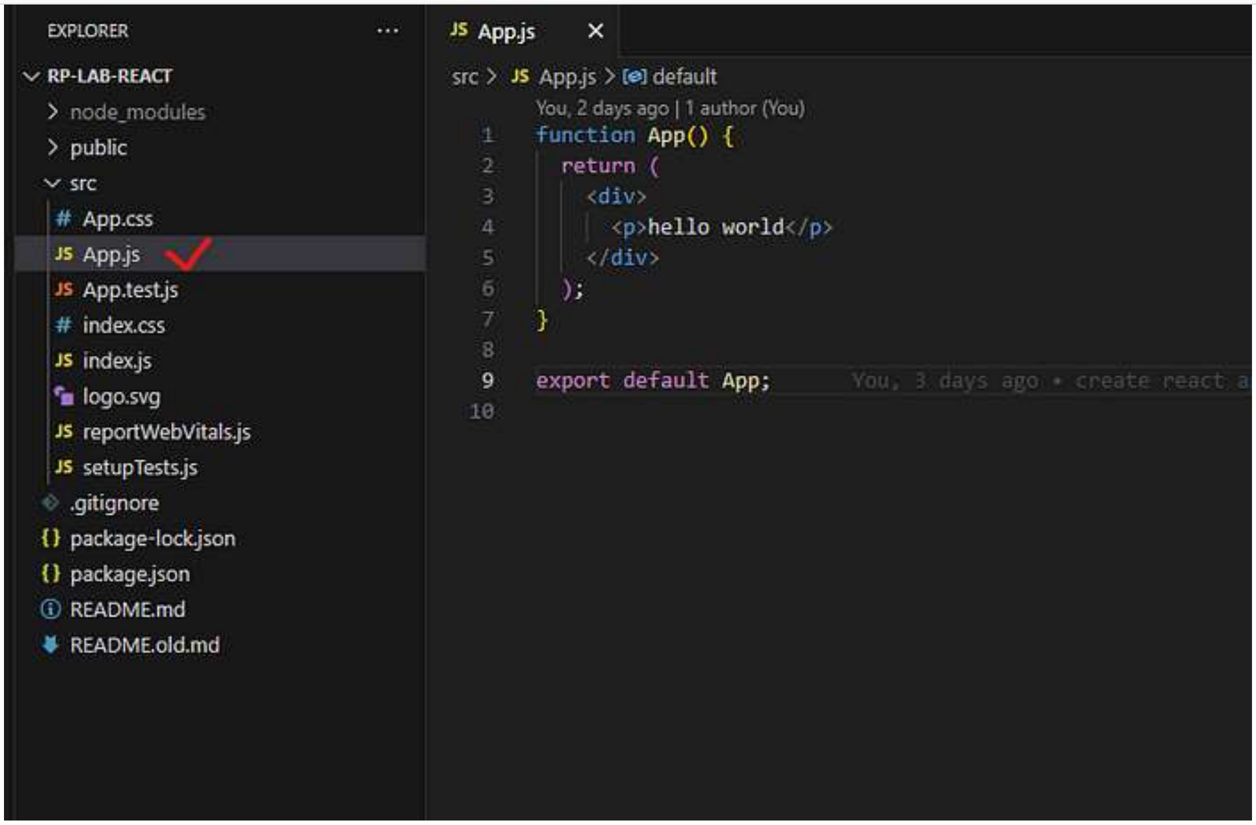
After successfully cloning the Git repository, open Visual Studio Code and navigate to the cloned repository folder. Then, create a React app using the following command:

```
npx create-react-app .
```

Make sure you have installed React and Node on your local machine.

Next, edit the `App.js` file to display a basic page that says "Hello, world."

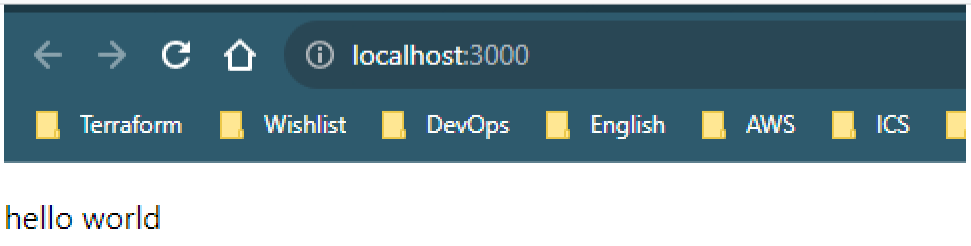
```
function App() {
  return (
    <div>
      <p>hello world</p>
    </div>
  );
}

export default App;
```



```
You, 2 days ago | 1 author (You)
function App() {
  return (
    <div>
      <p>hello world</p>
    </div>
  );
}

export default App;        You, 3 days ago • create react a
```

After the process is complete, let’s try running it locally on our machine first.

```
npm start
```



hello world

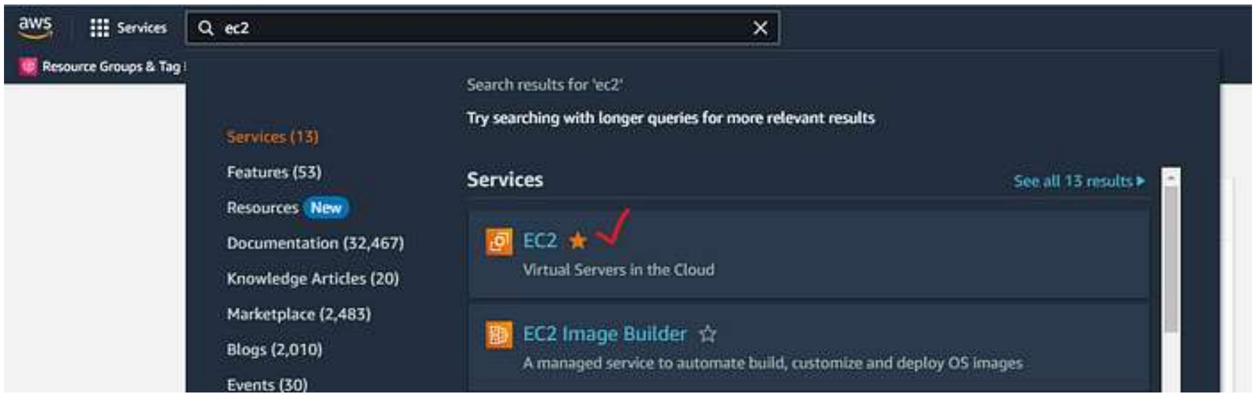
React app is working on local machine

Once our React app is ready, proceed to push the code to the GitHub repository.

```
git add .
git commit -am “create react app”
git push
```

2. Create EC2

Log in to your AWS account and search for EC2.



Search results for 'ec2'

Try searching with longer queries for more relevant results

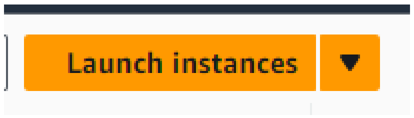
- Services (13)
- Features (53)
- Resources New
- Documentation (32,467)
- Knowledge Articles (20)
- Marketplace (2,483)
- Blogs (2,010)
- Events (30)

Services See all 13 results ▸

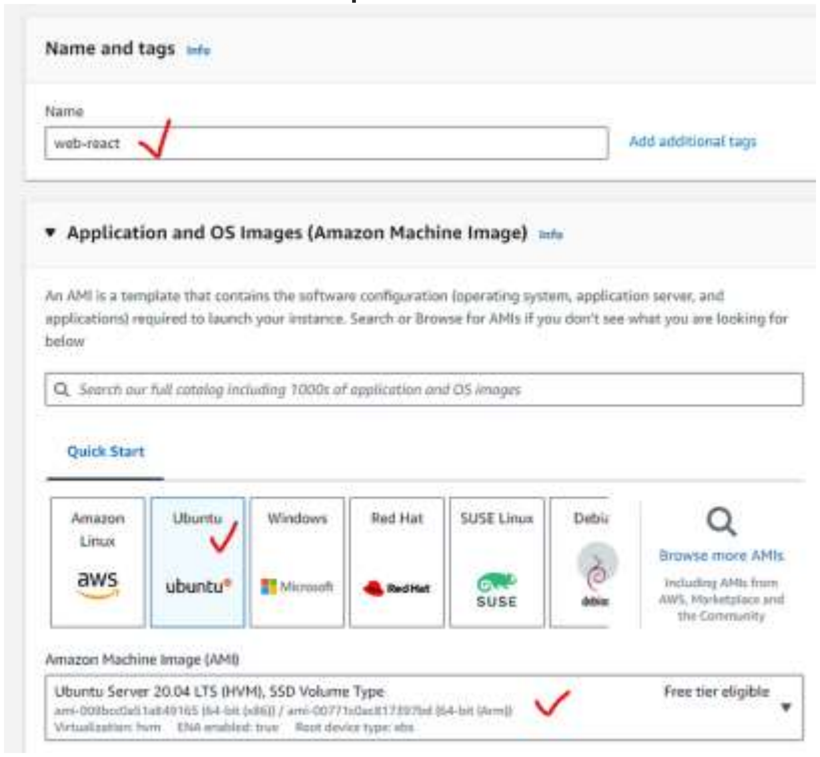
EC2 ★ ✓
Virtual Servers in the Cloud

EC2 Image Builder ☆
A managed service to automate build, customize and deploy OS images

From the EC2 console dashboard, in the **Launch instance** box, choose **Launch instance**



Enter a preferred name for your EC2 instance, and in the OS section, select Ubuntu 20.04.



Name and tags Info

Name
web-react ✓ Add additional tags

▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

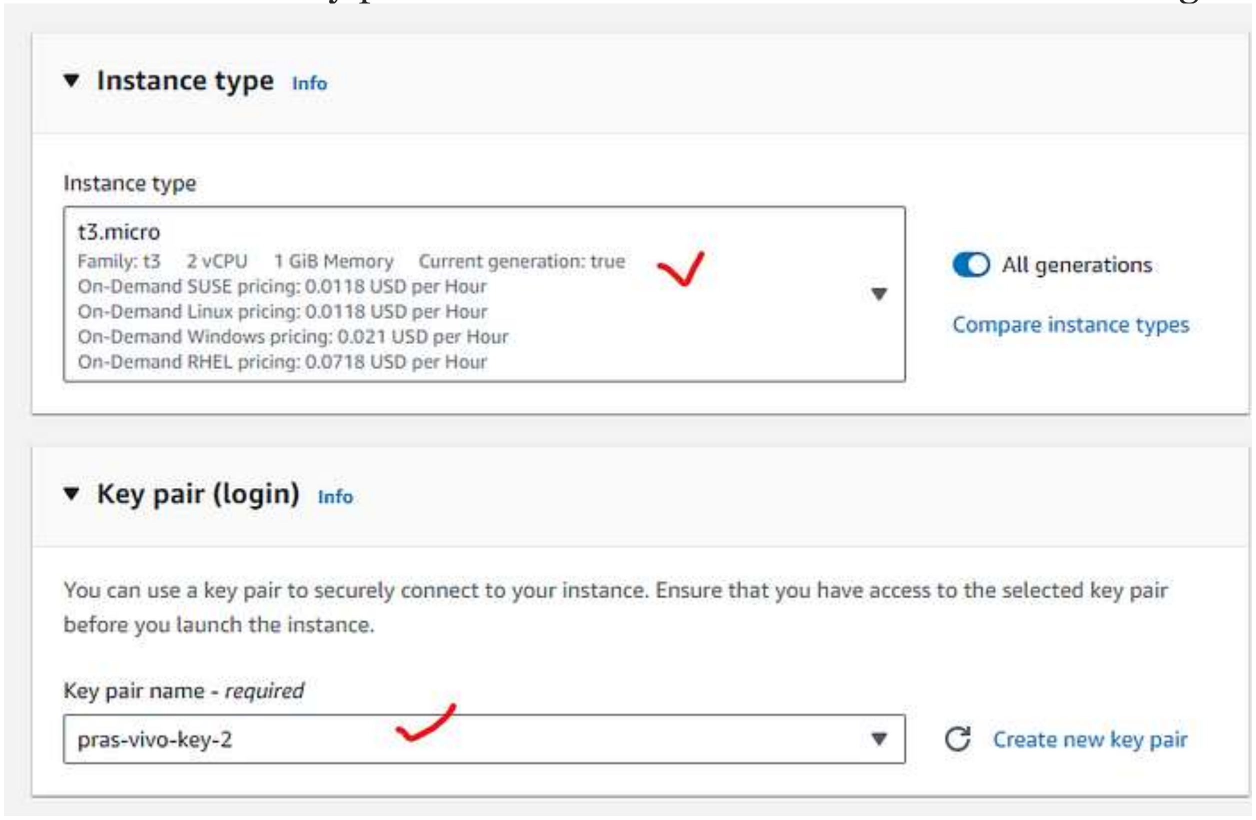
Quick Start

Amazon Linux | [✓ Ubuntu] | Windows | Red Hat | SUSE Linux | Debi

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)
Ubuntu Server 20.04 LTS (HVM), SSD Volume Type ✓ Free tier eligible
ami-008bcc0a51a849165 (64-bit (x86)) / ami-00771c0ac817397bd (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Select the t3.micro instance type and click on “Create new key pair.” We will use this key pair to SSH into the EC2 instance we’re creating.



▼ Instance type Info

Instance type

t3.micro
Family: t3 2 vCPU 1 GiB Memory Current generation: true
On-Demand SUSE pricing: 0.0118 USD per Hour
On-Demand Linux pricing: 0.0118 USD per Hour
On-Demand Windows pricing: 0.021 USD per Hour
On-Demand RHEL pricing: 0.0718 USD per Hour

All generations

Compare instance types

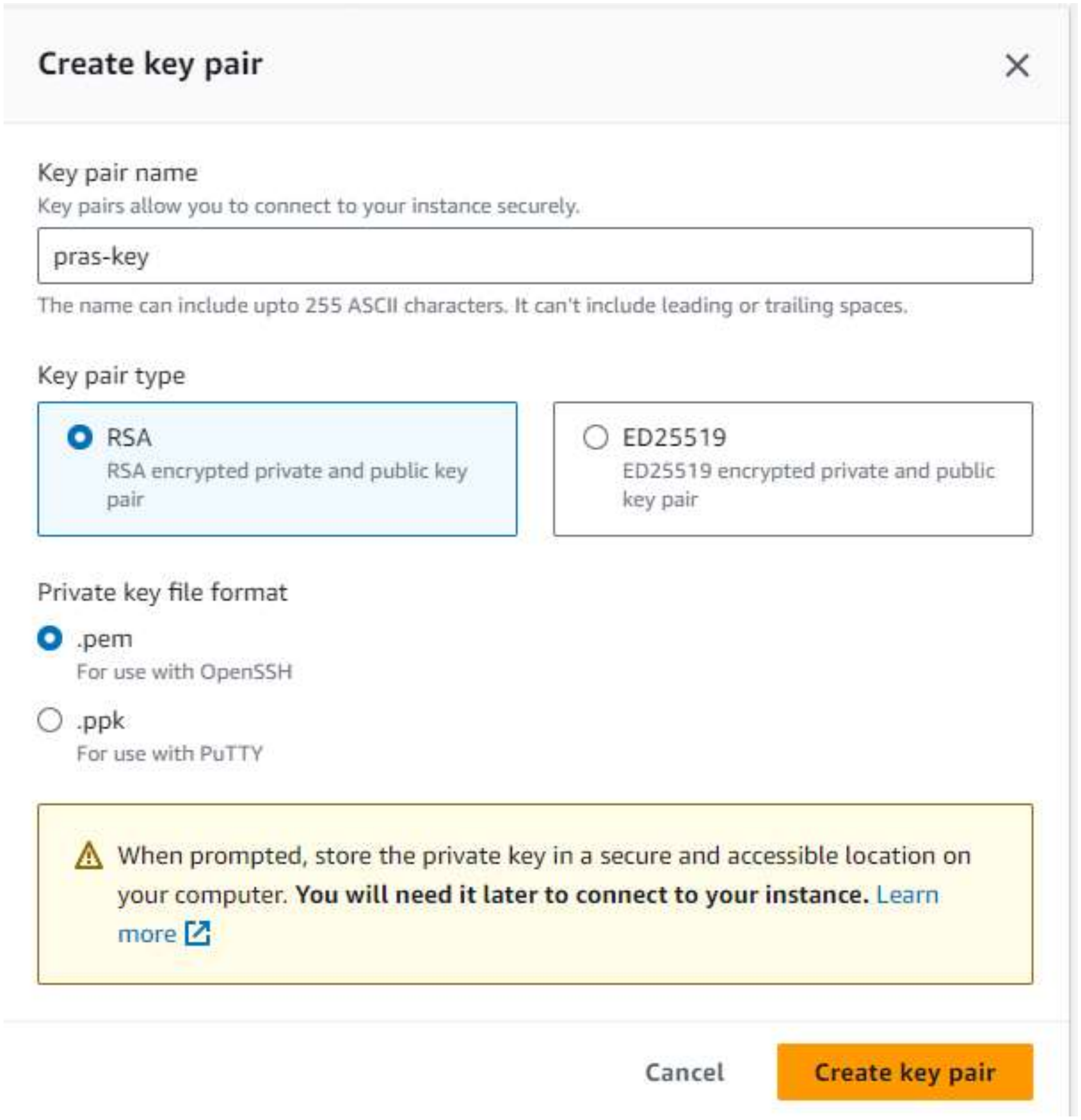
▼ Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

pras-vivo-key-2

Create new key pair



Create key pair

Key pair name

Key pairs allow you to connect to your instance securely.

pras-key

The name can include upto 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

[✓ RSA] RSA encrypted private and public key pair

ED25519 ED25519 encrypted private and public key pair

Private key file format

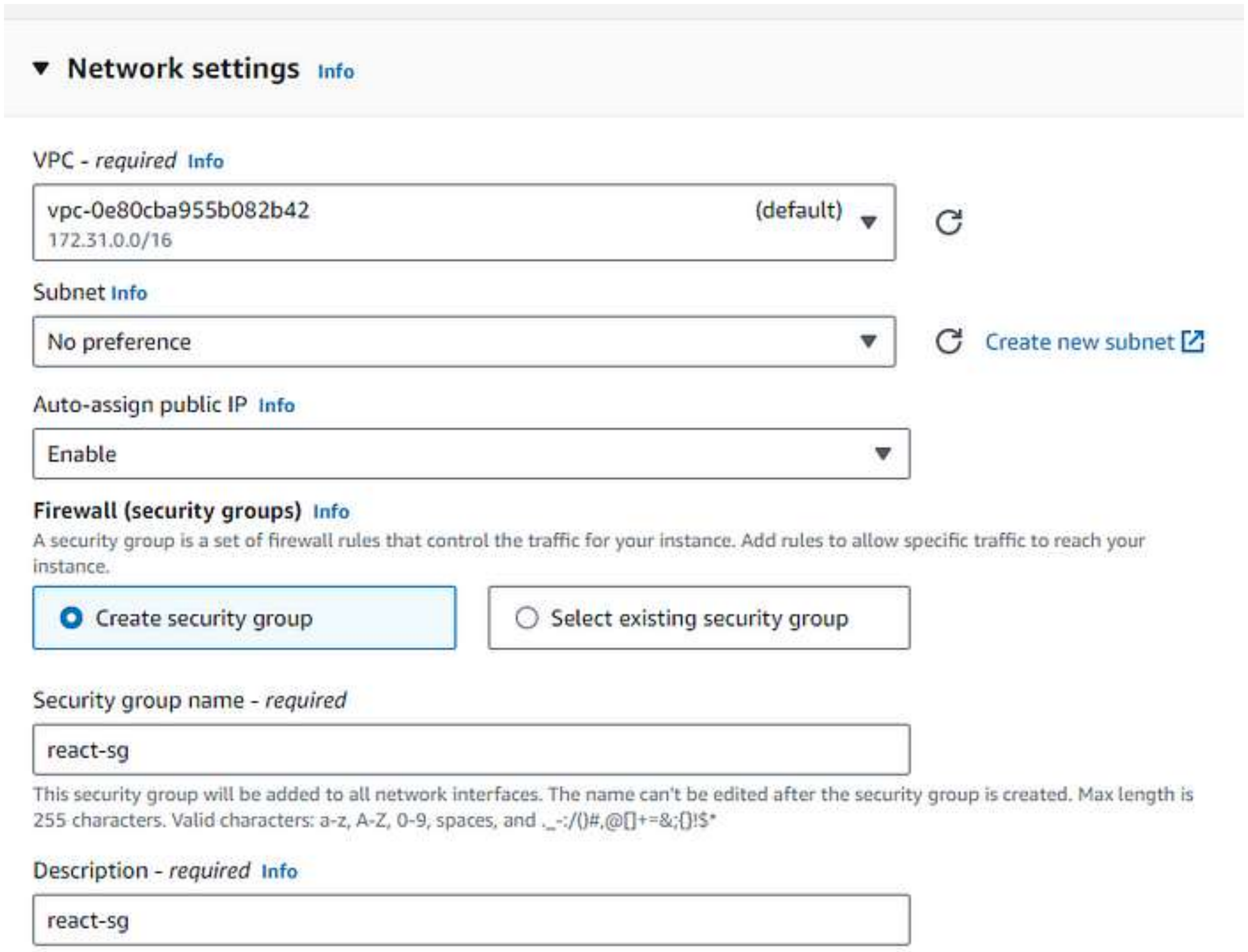
[✓ .pem] For use with OpenSSH

.ppk For use with PuTTY

When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn more

Give your key pair name and choose .pem. You can choose .ppk if you want use putty to SSH

Leave the VPC and subnet as default. Choose “Enable” for auto-assigning a public IP. Click on “Create security group” and provide a name for it.



▼ Network settings Info

VPC - *required* Info

vpc-0e80cba955b082b42 (default)
172.31.0.0/16

Subnet Info

No preference

Create new subnet

Auto-assign public IP Info

Enable

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

[✓ Create security group] Select existing security group

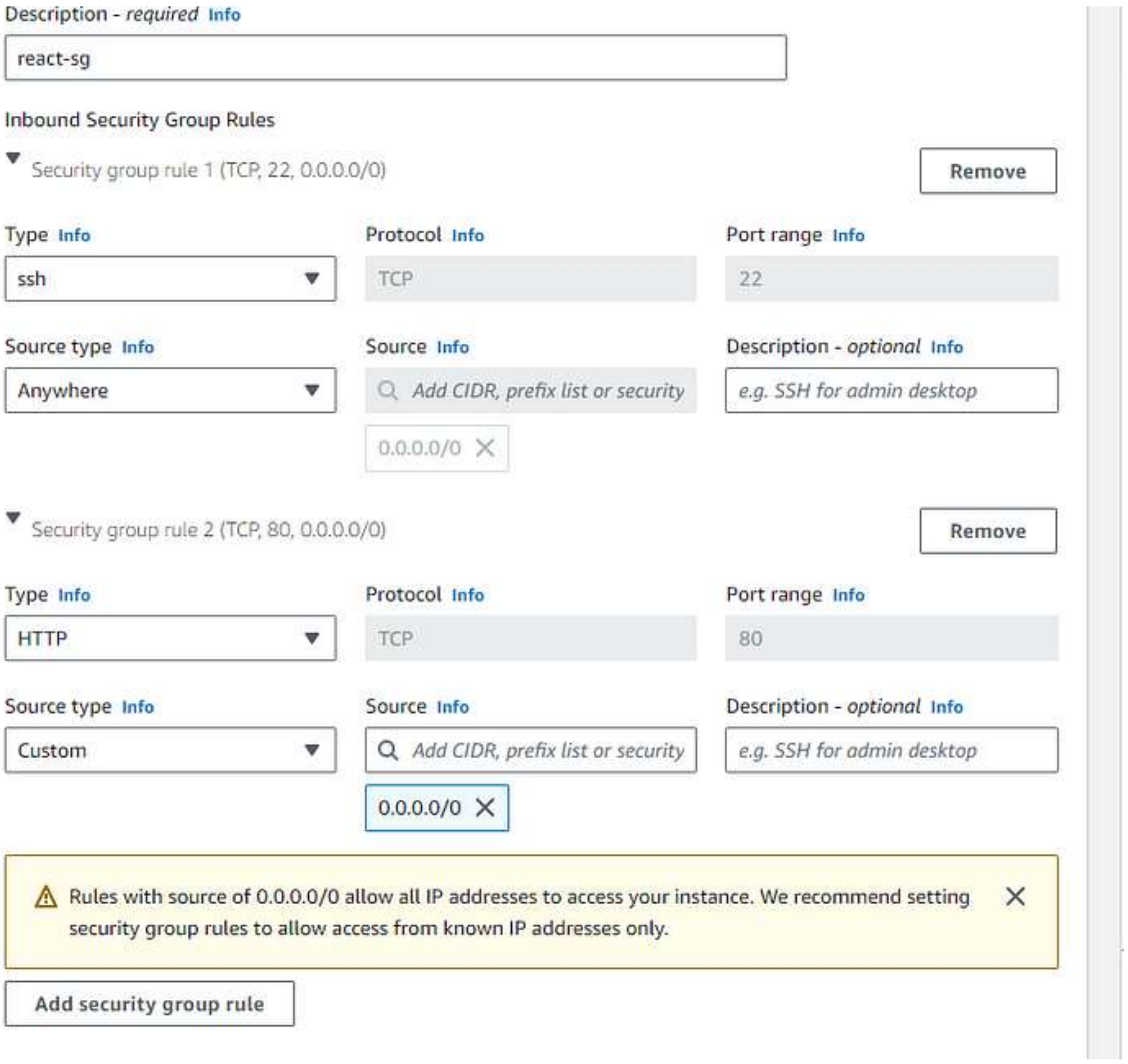
Security group name - *required*

react-sg

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*

Description - *required* Info

react-sg


Description - *required* Info

react-sg

Inbound Security Group Rules

Security group rule 1 (TCP, 22, 0.0.0.0/0) — Remove

Type	Protocol	Port range
ssh	TCP	22

Source type	Source	Description - *optional*
Anywhere	*Add CIDR, prefix list or security* 0.0.0.0/0	*e.g. SSH for admin desktop*

Security group rule 2 (TCP, 80, 0.0.0.0/0) — Remove

Type	Protocol	Port range
HTTP	TCP	80

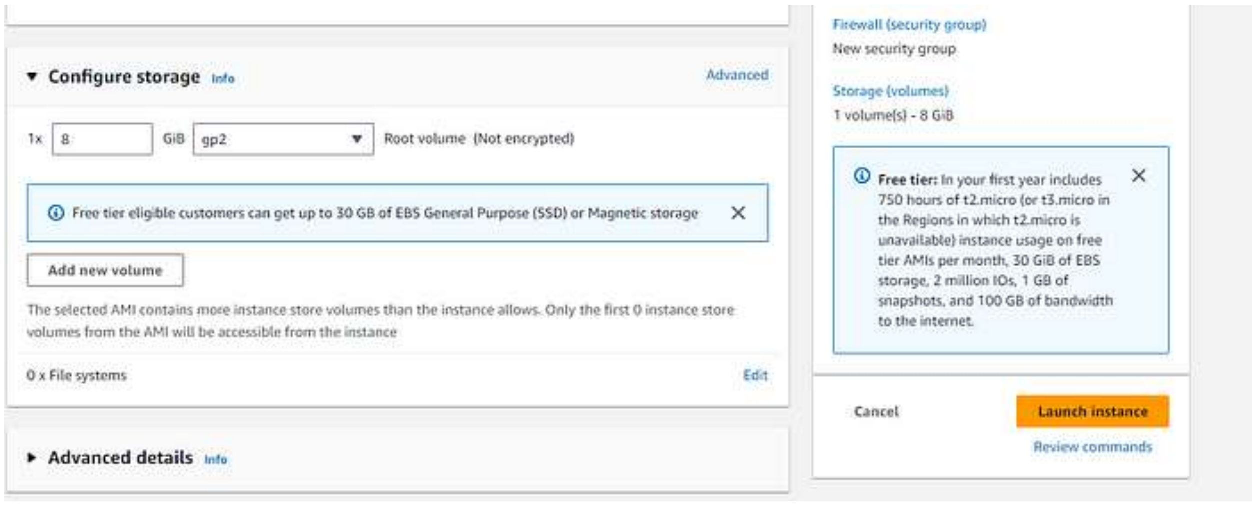
Source type	Source	Description - *optional*
Custom	*Add CIDR, prefix list or security* 0.0.0.0/0	*e.g. SSH for admin desktop*

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Add security group rule

Open SSH and HTTP to 0.0.0.0/0

Leave the rest of the configurations as default and click on "Launch Instance."


▾ Configure storage Info — Advanced

1x 8 GiB gp2 Root volume (Not encrypted)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

Add new volume

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

0 x File systems — Edit

▸ Advanced details Info

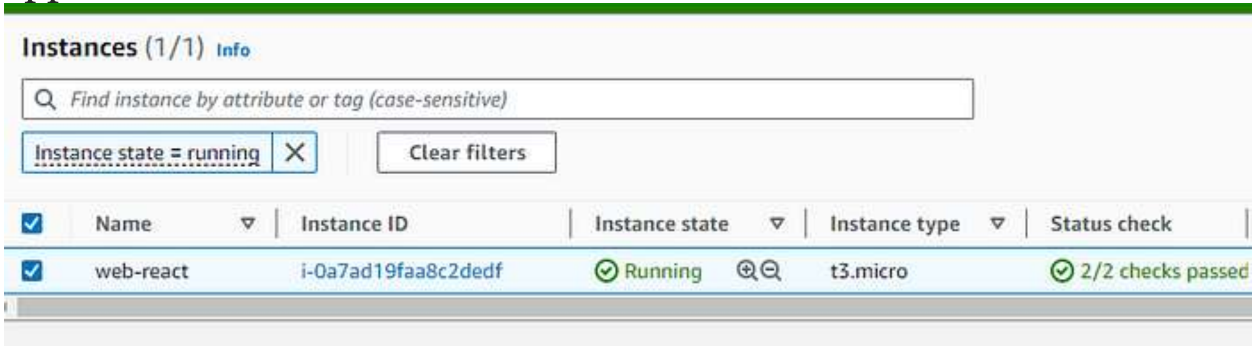
Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Cancel — Launch instance — Review commands

Wait until the instance status is "Running," then you can proceed to SSH into the EC2 instance to install the react dependency for our web app.


Instances (1/1) Info

Find instance by attribute or tag (case-sensitive)

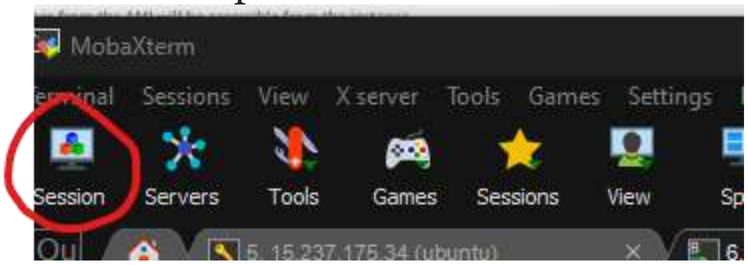
Instance state = running — Clear filters

☑	Name	Instance ID	Instance state	Instance type	Status check
☑	web-react	i-0a7ad19faa8c2dedf	✓ Running	t3.micro	✓ 2/2 checks passed

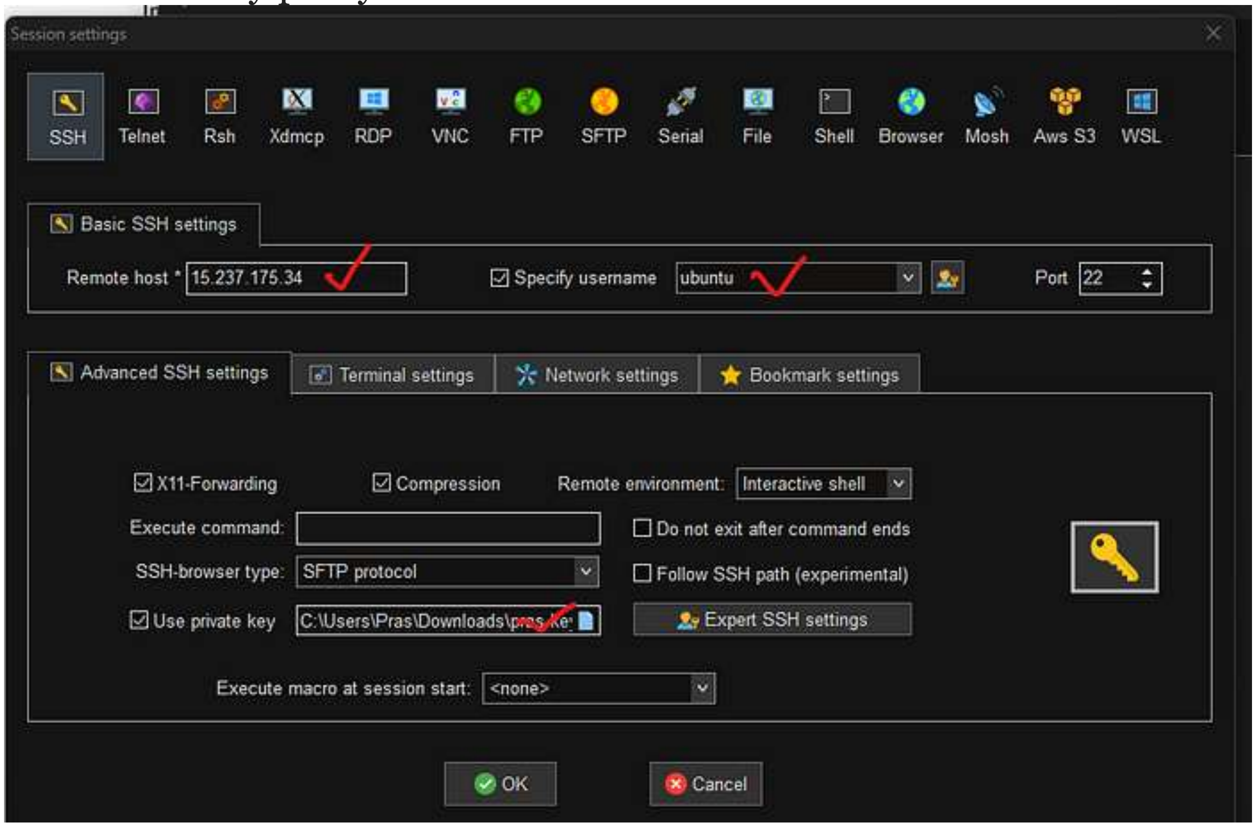
3. Install the dependency for react app

- To access the EC2 server, you can use MobaXterm, a free software that you can download from https://mobaxterm.mobatek.net/.
- If you've chosen a key pair in the .ppk format, you'll need to use PuTTY, which you can download from https://www.putty.org/.

- Open MobaXterm and select “Session.”


MobaXterm
erminal Sessions View X server Tools Games Settings
Session Servers Tools Games Sessions View Sp
5. 15.237.175.34 (ubuntu)

- Enter your EC2’s public IP address and use the username “ubuntu.” Check the “Use private key” option and select the key pair you created in AWS.



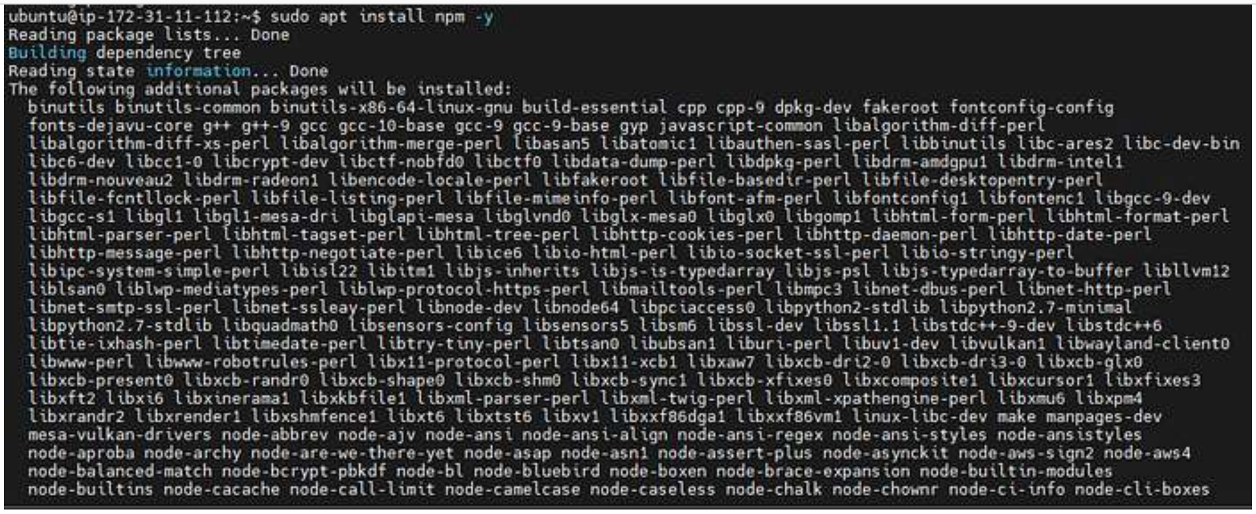
Remote host * 15.237.175.34
☑ Specify username ubuntu
Port 22
☑ X11-Forwarding
☑ Compression
Remote environment: Interactive shell
Execute command:
☐ Do not exit after command ends
SSH-browser type: SFTP protocol
☐ Follow SSH path (experimental)
☑ Use private key C:\Users\Pras\Downloads\pras.ke
Execute macro at session start: <none>

- Update the package

```
sudo apt-get update -y
```

- Install npm

```
sudo apt install npm -y
```



```
ubuntu@ip-172-31-11-112:~$ sudo apt install npm -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential cpp cpp-9 dpkg-dev fakeroot fontconfig-config
  fonts-dejavu-core g++ g++-9 gcc gcc-10-base gcc-9 gcc-9-base gyp javascript-common libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan5 libatomic1 libauthen-sasl-perl libbinutils libc-ares2 libc-dev-bin
  libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0 libctf0 libdata-dump-perl libdpkg-perl libdrm-amdgpu1 libdrm-intel1
  libdrm-nouveau2 libdrm-radeon1 libencode-locale-perl libfakeroot libfile-basedir-perl libfile-desktopentry-perl
  libfile-fcntllock-perl libfile-listing-perl libfile-mimeinfo-perl libfont-afm-perl libfontconfig1 libfontenc1 libgcc-9-dev
  libgcc-s1 libgl1 libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgomp1 libhtml-form-perl libhtml-format-perl
  libhtml-parser-perl libhtml-tagset-perl libhtml-tree-perl libhttp-cookies-perl libhttp-daemon-perl libhttp-date-perl
  libhttp-message-perl libhttp-negotiate-perl libice6 libio-html-perl libio-socket-ssl-perl libio-stringy-perl
  libipc-system-simple-perl libisl22 libitm1 libjs-inherits libjs-is-typedarray libjs-psl libjs-typedarray-to-buffer libllvm12
  liblsan0 liblwp-mediatypes-perl liblwp-protocol-https-perl libmailtools-perl libmpc3 libnet-dbus-perl libnet-http-perl
  libnet-smtp-ssl-perl libnet-ssleay-perl libnode-dev libnode64 libpciaccess0 libpython2-stdlib libpython2.7-minimal
  libpython2.7-stdlib libquadmath0 libsensors-config libsensors5 libsm6 libssl-dev libssl1.1 libstdc++-9-dev libstdc++6
  libtie-ixhash-perl libtimedate-perl libtry-tiny-perl libtsan0 libubsan1 liburi-perl libuv1-dev libvulkan1 libwayland-client0
  libwww-perl libwww-robotrules-perl libx11-protocol-perl libx11-xcb1 libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0
  libxcb-present0 libxcb-randr0 libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcomposite1 libxcursor1 libxfixes3
  libxft2 libxi6 libxinerama1 libxkbfile1 libxml-parser-perl libxml-twig-perl libxml-xpathengine-perl libxmu6 libxpm4
  libxrandr2 libxrender1 libxshmfence1 libxt6 libxtst6 libxv1 libxxf86dga1 libxxf86vm1 linux-libc-dev make manpages-dev
  mesa-vulkan-drivers node-abbrev node-ajv node-ansi node-ansi-align node-ansi-regex node-ansi-styles node-ansistyles
  node-aproba node-archy node-are-we-there-yet node-asap node-asn1 node-assert-plus node-asynckit node-aws-sign2 node-aws4
  node-balanced-match node-bcrypt-pbkdf node-bl node-bluebird node-boxen node-brace-expansion node-builtin-modules
  node-builtins node-cacache node-call-limit node-camelcase node-caseless node-chalk node-chownr node-ci-info node-cli-boxes
```

Installing npm

- Install node version 20

```
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
sudo apt install -y nodejs
```

- Install the Nginx web server to run your react

```
sudo apt install nginx -y
```

- Create a directory for react

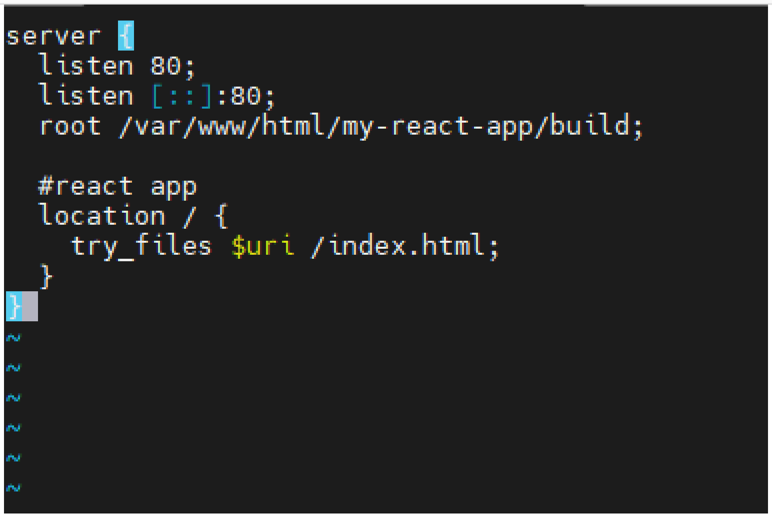
```
sudo mkdir /var/www/html/my-react-app
```

- Configure Nginx

```
sudo vi /etc/nginx/conf.d/react.conf
```

- Update the server block

```
server {
  listen 80;
  listen [::]:80;
  root /var/www/html/my-react-app/build;

  #react app
  location / {
    try_files $uri /index.html;
  }
}
```



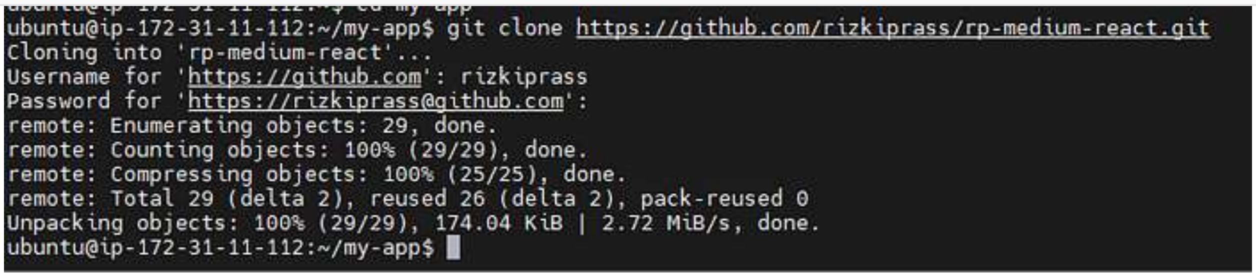
```
server {
  listen 80;
  listen [::]:80;
  root /var/www/html/my-react-app/build;

  #react app
  location / {
    try_files $uri /index.html;
  }
}
~
~
~
~
~
~
```

- Create a folder named “my-app” to place your React app project.

```
cd /home/ubuntu
mkdir my-app
cd my-app
```

- Clone react app

```
git clone <repo-url>
```



```
ubuntu@ip-172-31-11-112:~/my-app$ git clone https://github.com/rizkiprass/rp-medium-react.git
Cloning into 'rp-medium-react'...
Username for 'https://github.com': rizkiprass
Password for 'https://rizkiprass@github.com':
remote: Enumerating objects: 29, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 29 (delta 2), reused 26 (delta 2), pack-reused 0
Unpacking objects: 100% (29/29), 174.04 KiB | 2.72 MiB/s, done.
ubuntu@ip-172-31-11-112:~/my-app$
```

I change the repository name from rprass-react-app to rp-medium-react. Sorry for the confusion

If you're prompted for a password, follow this link to create a Personal Access Token (PAT)

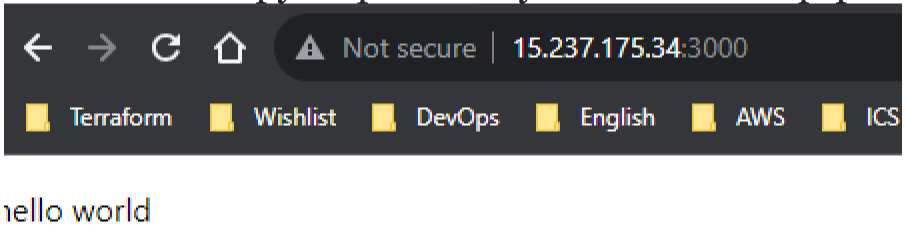
- Install your React project dependency

```
cd <project-folder>
npm install
```

- Test the React app first to ensure it's functioning correctly.

```
npm start
```

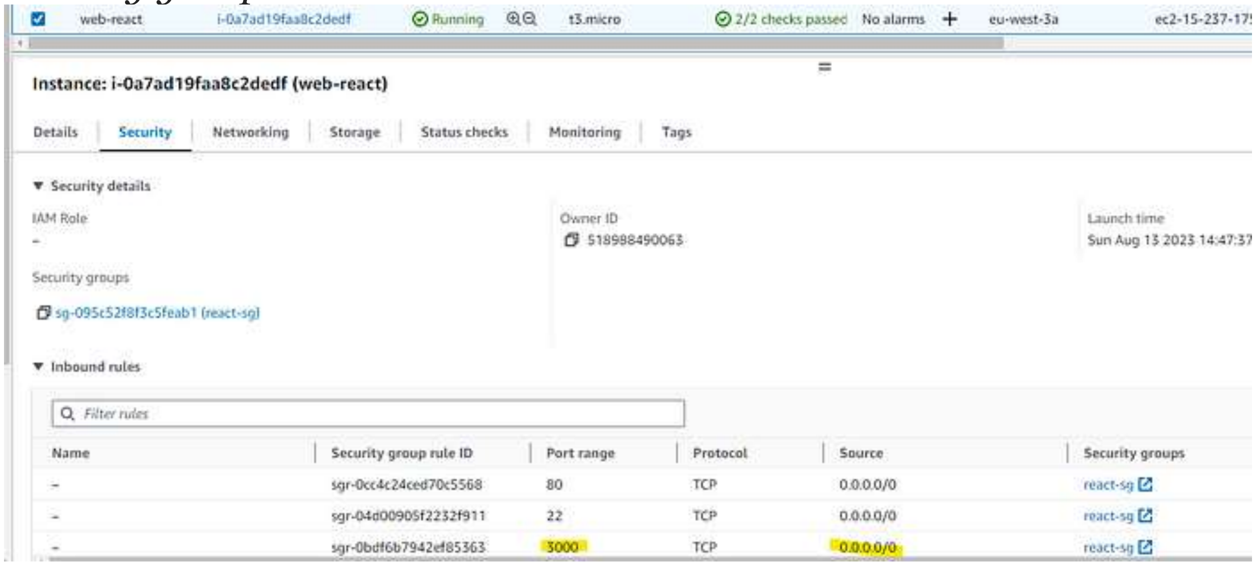
- Copy IP public to your browser <ip-public>:3000



ıello world

The React app is running in development mode.

if the page show error/not found, make sure you open port 3000 at security group



web-react i-0a7ad19faa8c2dedf Running t3.micro 2/2 checks passed No alarms eu-west-3a ec2-15-237-17

Instance: i-0a7ad19faa8c2dedf (web-react)

▼ Security details

IAM Role
-

Owner ID
518988490063

Launch time
Sun Aug 13 2023 14:47:37

Security groups
sg-095c52f8f3c5feab1 (react-sg)

▼ Inbound rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups
-	sgr-0cc4c24ced70c5568	80	TCP	0.0.0.0/0	react-sg
-	sgr-04d00905f2232f911	22	TCP	0.0.0.0/0	react-sg
-	sgr-0bdf6b7942ef85363	3000	TCP	0.0.0.0/0	react-sg

Open port 3000 to anywhere (0.0.0.0/0)

- Build React app

```
npm run build
```

- Copy the “build” folder to the “/var/www/html” directory so that Nginx can read from this folder.

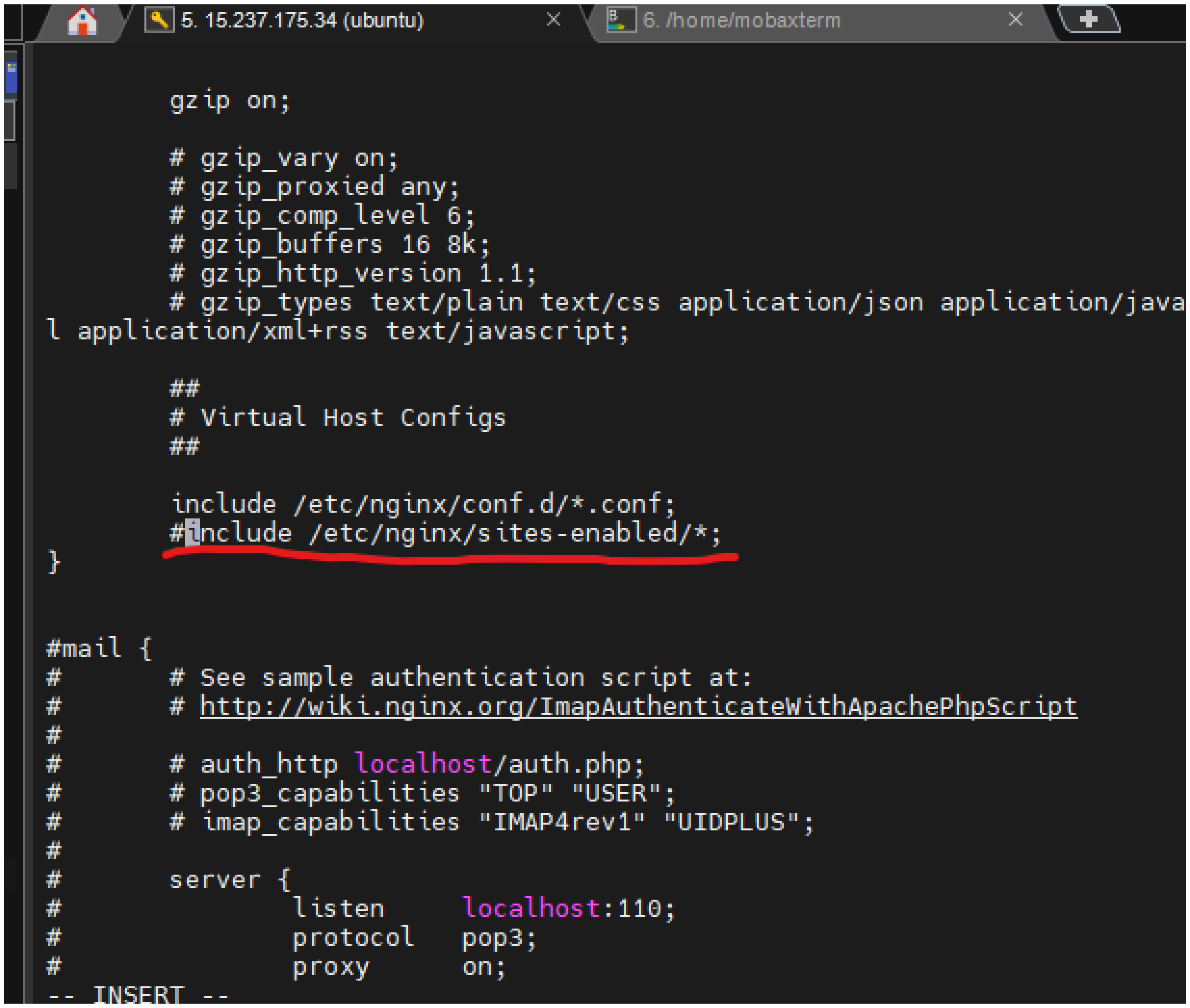
```
sudo cp -R build/ /var/www/html/my-react-app/
```

- Disable the nginx default configuration

```
sudo vi /etc/nginx/nginx.conf
```

comment below line using “#”:

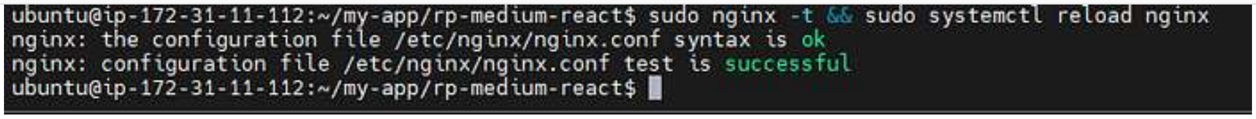
```
#include /etc/nginx/sites-enabled/*;
```



```
        gzip on;

        # gzip_vary on;
        # gzip_proxied any;
        # gzip_comp_level 6;
        # gzip_buffers 16 8k;
        # gzip_http_version 1.1;
        # gzip_types text/plain text/css application/json application/java
l application/xml+rss text/javascript;

        ##
        # Virtual Host Configs
        ##

        include /etc/nginx/conf.d/*.conf;
        #include /etc/nginx/sites-enabled/*;
}


#mail {
#       # See sample authentication script at:
#       # http://wiki.nginx.org/ImapAuthenticateWithApachePhpScript
#
#       # auth_http localhost/auth.php;
#       # pop3_capabilities "TOP" "USER";
#       # imap_capabilities "IMAP4rev1" "UIDPLUS";
#
#       server {
#               listen     localhost:110;
#               protocol   pop3;
#               proxy      on;
-- INSERT --
```

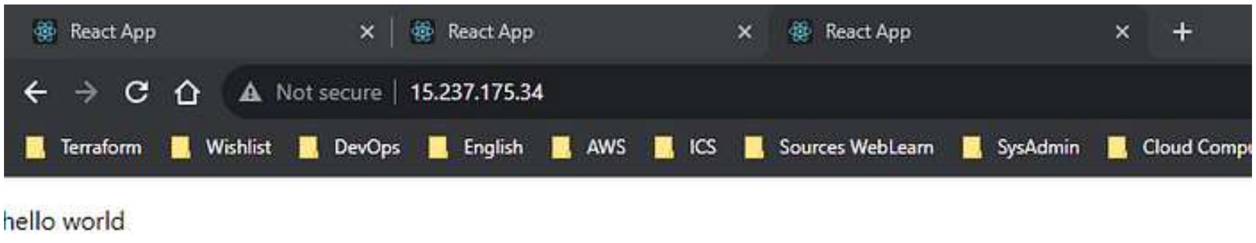
- Validate the nginx configuration and reload the nginx

```
sudo nginx -t && sudo systemctl reload nginx
```



```
ubuntu@ip-172-31-11-112:~/my-app/rp-medium-react$ sudo nginx -t && sudo systemctl reload nginx
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
ubuntu@ip-172-31-11-112:~/my-app/rp-medium-react$
```

- Copy the public IP of your EC2 instance and paste it into your browser. You should now be able to access your React app on port 80, as we are using the Nginx web server.



hello world

Your react app now deploy on AWS EC2

Congratulations, you have successfully deployed a React app on EC2 and made it accessible to all users.