summit

Black Lung Dashboard
Amazon Web Services Deployment
Technical Requirements and Guide

Date:	September 2023
Prepared for:	U.S. Department of Labor Contract Number: 1605DC-18-R-00044
Prepared by:	Summit Consulting, LLC 777 6th St. NW, Suite 520 Washington, DC 20001 | www.summitllc.us

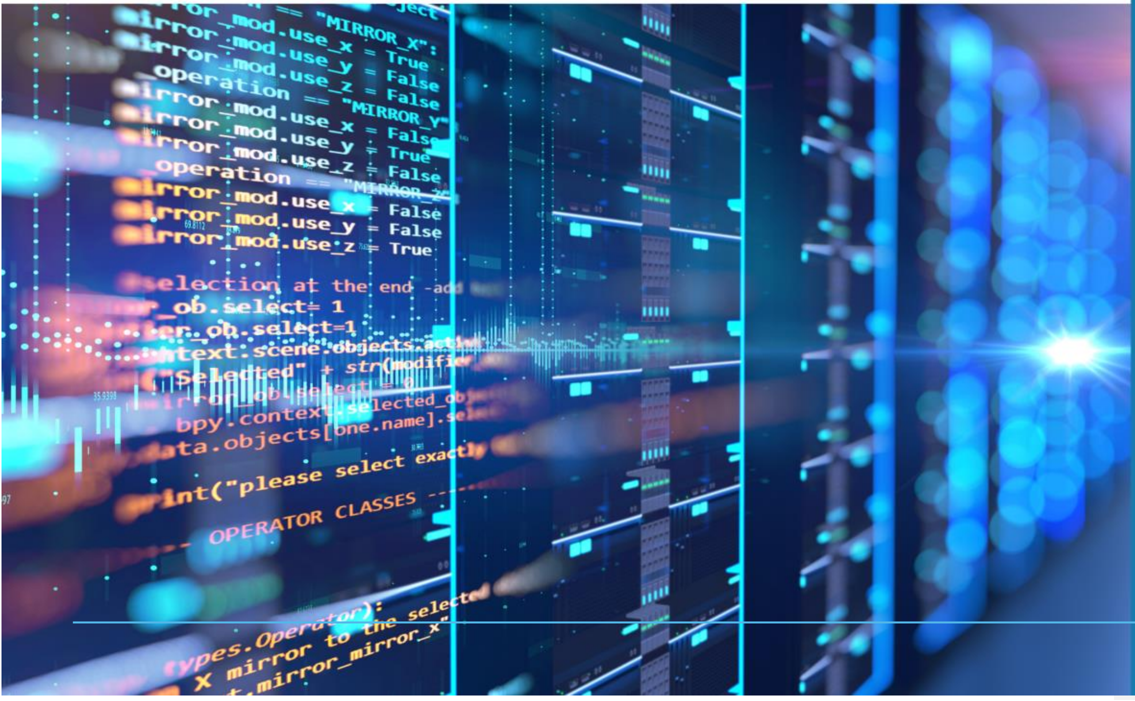


Table of contents

1 What Technology Stack MSHA will need

- R
  - Version 4.2.1
- Amazon Web Services (AWS)
  - Set up EC2 Instance (Ubuntu 20.04).
  - Install and configure shiny-server.
  - Install and configure nginx.
- A Github account
  - Summit will grant access for MSHA to clone our current repository and have all code/application history, including README documentation.

2 Setting Up an AWS EC2 Instance

1. Login to the AWS account where we will host the application (app).
2. Navigate to the EC2 Dashboard (Services -> EC2).
3. Click on Launch Instance to launch a new instance and go through the configuration steps.
    a. Confirm your region is the desired region (most likely `N. Virginia, us-east-1`).
    b. Choose an Ubuntu machine (>=20.04) and AMI.
    c. Choose an Instance Type.
        i. We recommend `t2.small` if less than 25 users will access the app or a `t3.medium` if more than 25 users will access the app.
    d. Choose Instance Details (default should be fine).
    e. Add storage.
        i. We will later install ~ 10GB of software.
        ii. You will need enough disk space to accommodate this and your application.
    f. Add tags.
        i. Add a name tag.
            (1) Click Add Tag.
            (2) Enter `Name` in the key field.
            (3) Enter a concise description of your app in the `Value` field.
        ii. Add any other tags you may want.
    g. Configure Security Groups.
        i. SSH (Port 22) access from your company's VPN (Ask Information Technology (IT) for IP)
        ii. HTTP (Port 80)
            (1) Probably the same.
            (2) If a public facing application, use (0.0.0.0:/0).
        iii. HTTPS (Port 443)
            (1) Probably the same.
            (2) If a public facing application, use (0.0.0.0:/0).
4. Review Instance Launch and Launch.
5. Create two new AWS key pairs and save the private keys in a secure location. You will need this to access your machine in later steps. One file will end in .pem and the other will end in .ppk
6. Allocate an Elastic IP to the EC2 Instance.
    a. Under Network and Security (left side panel), click on Elastic IPs.
    b. Click the Action drop-down, and then Allocate new address.
    c. Select the Elastic IP, click Add/Edit Tags to add a `Name` tag describing its association with the shiny application.
    d. Select your new instance in the Instance field and click Associate.
7. If hosting this app on https and not just localhost, contact your IT team to create a domain name to associate with your app's Elastic IP. You will need to provide IT with at least:
    a. The domain name you want your app to use (e.g., this-app.agency.gov).
    b. The Elastic IP you just associated with your AWS EC2.

3 SSH Capability

3.1 Connecting to your Instance via SSH

1. Open Putty.
2. Type Your host Name.
    a. Should be `ubuntu@your.elatic.ip.address`.
    b. Make sure the port is 22, and the connection type is SSH.
3. Click on the SSH dropdown on the left-hand side, and then click Auth.
4. Browse for your "Private key file for authentication".
    a. This will be the key pair saved earlier that ends in .ppk.

3.2 Creating a new user and new SSH key

All of the information can be found here, although there are revisions below:

3.2.1 Creating a new SSH key

1. Open PuttyGen.
2. Press Generate button and move mouse around area.
3. Once the keys are generated, type your key passphrase (if you want one).
4. Save key comment as your.email@summitllc.us.
5. Save Public key. (username.pub.key).
6. Save Private key (username.ppk).

3.2.2 Assigning the SSH key to a new user (skip steps 1 and 2 if user already exists)

1. Type `adduser usernamehere`.
2. Fill in the required details for user set-up.
3. Type the following commands:

```
su usernamehere
cd /home/usernamehere
mkdir .ssh
chmod 700 .ssh
cd .ssh
```

4. Create and edit a file called authorized_keys.

```
vi authorized_keys
i
```

5. Copy/paste your public ssh key, on ONE LINE (right click to paste).
   Do not add the your.email@summitllc.us at the end of the line.
   Do not add the **BEGIN PUBLIC KEY** or **END PUBLIC KEY**.
   Do not add the **rsa-key-20090614** at the end.
   Make sure, there is **ssh-rsa** at the beginning.

```
ESC :x (to save and exit)
```

6. Final command:

```
Chmod 600 authorized_keys
```

3.2.3 Adding the user to sshd_config

1. Type following commands:

```
cd etc/ssh
sudo vi sshd_config
```

2. Add your username to the list and save file
3. Then type:

```
sudo service ssh restart
```

4. Login to Putty as your username with your ssh key.

4 Installing Software and Packages

4.1 Install R and Shiny Server

Follow the instructions here:

```
sudo apt-get update
sudo apt-get install r-base
sudo su - \ -c "R -e \"install.packages('shiny',
repos='https://cran.rstudio.com/')\""

sudo apt-get install gdebi-core

wget https://download3.rstudio.org/ubuntu-
18.04/x86_64/shiny-server-1.5.20.1002-amd64.deb

sudo gdebi shiny-server-1.5.20.1002-amd64.deb
```

4.2 Install System Libraries

```
sudo apt-get -y install \
    default-jdk \
    libcurl4-gnutls-dev \
    libpq-dev \
    libssl-dev \
    libxml2-dev \
    libcairo2-dev \
    libgdal-dev \
    libglpk-dev \
    libgpgme11-dev \
    libproj-dev \
    libudunits2-dev \
    libv8-dev \
    texlive-full \
    unixodbc-dev
```

4.3 Install Certbot and nginx

```
sudo apt-get -y install nginx
sudo apt-get -y install software-properties-common
sudo add-apt-repository -y ppa:certbot/certbot
sudo apt-get -y update
sudo apt-get -y install python3-certbot-nginx
```

4.4 Installing PostgreSQL

Run the script postgresql.sh or its contents:

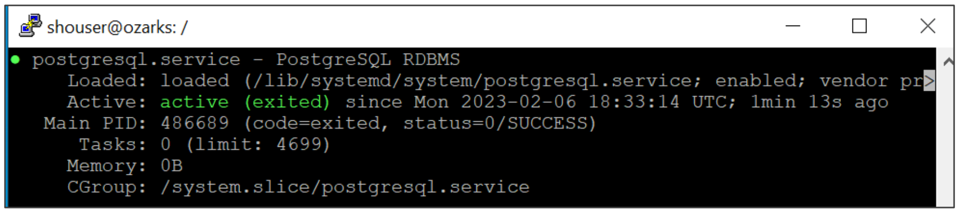
```
sudo apt-get update
```

```
# Install package ------------------------------------
----------------
sudo apt install postgresql postgresql-contrib
# Ensure service is started
sudo systemctl start postgresql.service
```

Ensure that it is active with the following command:

```
service postgresql status
```

If it is active, the output should look like this:



```
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor pr>
     Active: active (exited) since Mon 2023-02-06 18:33:14 UTC; 1min 13s ago
   Main PID: 486689 (code=exited, status=0/SUCCESS)
      Tasks: 0 (limit: 4699)
     Memory: 0B
     CGroup: /system.slice/postgresql.service
```

For more information visit How to install postgreSQL.

5 Configuring SSL Encryption

5.1 Configure Shiny Server

```
sudo systemctl stop shiny-server
sudo nano /etc/shiny-server/shiny-server.conf
```

Make edits if you would like and then:

```
sudo systemctl start shiny-server
```

5.2 Reverse Proxy

5.2.1 Configure new shiny configuration file

```
sudo service nginx stop
cd /etc/nginx
cd sites-available
sudo nano shiny.conf
```

This is for localhost apps:

```
server {
    # Listen on 80 port
    listen 80;
    # For IPv6 addresses
    listen [::]:80;
    # The reverse proxy
    location / {
        proxy_pass http://127.0.0.1:3838/;
        proxy_redirect http://127.0.0.1:3838/ $scheme://$host/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $connection_upgrade;
        proxy_read_timeout 20d;
        proxy_buffering off;
    }
}
```

This is for domain name hosted apps:

```
server {
    # Listen on 80 port
    listen 80;
    # For IPv6 addresses
    listen [::]:80;
    # Server name
    server_name your_domain_name;
    # The reverse proxy
    location / {
        proxy_pass http://127.0.0.1:3838/;
        proxy_redirect http://127.0.0.1:3838/ $scheme://$host/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $connection_upgrade;
        proxy_read_timeout 20d;
        proxy_buffering off;
    }
}
```

5.2.2 *Create Symbolic link and configure nginx*

```
cd /etc/nginx/sites-enabled
sudo ln -s /etc/nginx/sites-available/shiny.conf
/etc/nginx/sites-enabled/
sudo nano /etc/nginx/nginx.conf
```

Add the following in the http block in the nginx configuration file:

```
map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
}
```

5.2.3 *Remove default symbolic link*

```
cd /etc/nginx/sites-enabled
sudo rm default
cd /etc/nginx/sites-available
sudo rm default
```

5.2.4 *Restart nginx*

```
sudo service nginx start
```

6 Deploying Your App

6.1 Remove Default Shiny Files

```
sudo chmod 7777 /srv/shiny-server/
sudo rm /srv/shiny-server/index.html
sudo rm -rf /srv/shiny-server/sample-apps
```

6.2 Add app files to /srv/shiny-server/

This can be done manually with WinSCP.

6.3 Install necessary R packages

1. Login to putty as a user with sudoers privileges (i.e. with the AWS ppk).
2. Enter R for the terminal: sudo -i R.
3. Install all the packages you need for your shiny app: install.packages().

7 Modifying Your App

You will need someone familiar with R to make changes to the application. The app consists of the following files:

- Preprocessing_map_data.R
- UI Files
- Server Files
- global.R

7.1 Preprocessing_map_data.R

This file is intended for data preprocessing. It takes the raw data in an excel format and combines it with counties from the 2020 census for mapping. It cleans the data and then packages it in an RDS format for better application performance.

7.2 UI Files

7.2.1 ui.R

Here we set up the major framework of the application, as well as call all the helper ui files discussed in more detail below. We also include all css code within the ui.R file for easy app customization.

7.2.2 Others

We have modularized the application UI code into 5 additional files: ui_data_dict.R, ui_data_disclosures.R, ui_home.R, ui_map.R, and ui_time_series.R. These files set up the structure of each of their namesake tabs within the application.

7.3 Server Files

7.3.1 server.R

In server.R we call our five helper backend files detailed below.

7.3.2 Others

Our server files follow the same naming convention as their corresponding UI files giving server_data_dict.R, server_data_disclosures.R, server_home.R, server_map.R, and server_time_series.R. These files contain a majority of the content and reactivity of the black lung Shiny app. In each of these files you can find the respective code for their data tables, html, value boxes, leaflet maps, and ggplot visualizations.

Commented [SH1]: shannon to rename time_series to additional_graphs

7.4 global.R

Within global.R, we load our data and define our global variables and functions to be used throughout the application.