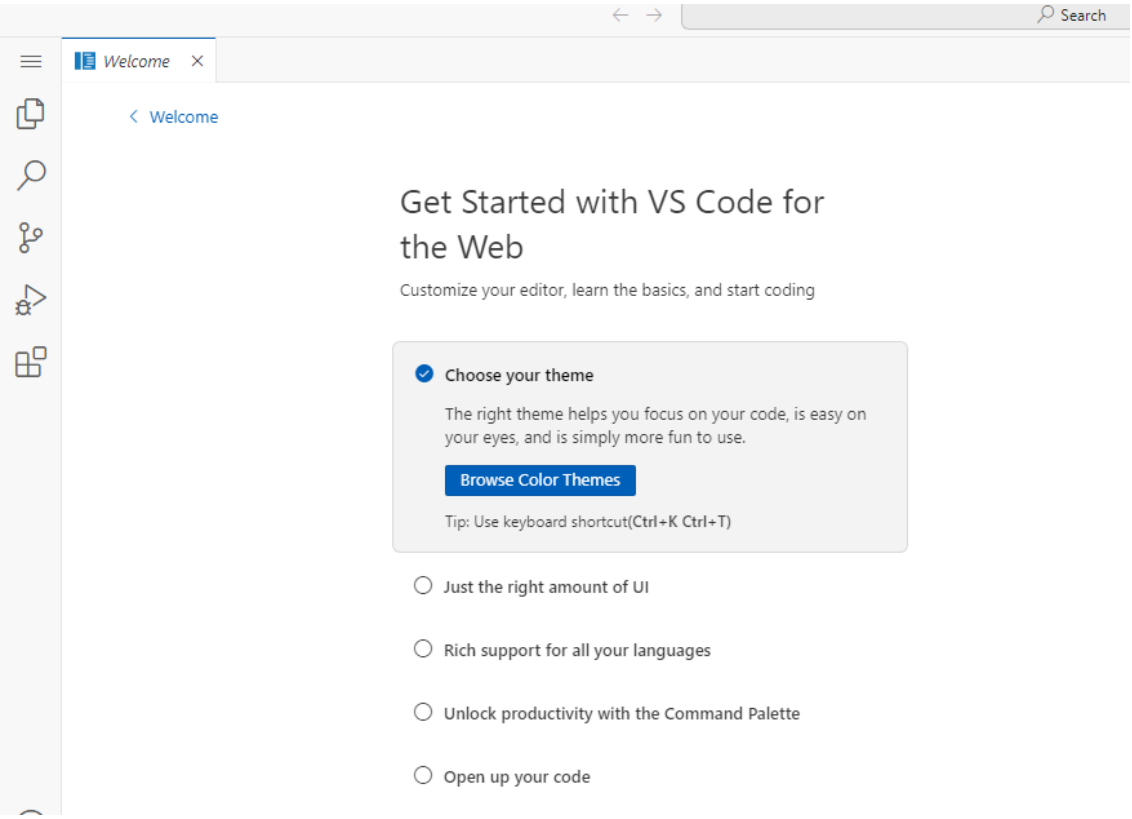
Usage instructions:

1. Launch the product via 1-click. **Please wait until** the instance passes **all** status checks and is running in the AWS console.

2. Once the instance is completely running you can access the code-server GUI. Open a **web browse**r and navigate to: **http://your_instance_public _ip:8081**

- **For ex: http://32.05.62:8081**
- For help: https://code.visualstudio.com/docs/getstarted/userinterface



Access the terminal prompt from the left navigation tabs

Helpful Starter Information

If VS Code Server is not running, log into your instance. You can connect using your Amazon private key and '**ubuntu**' login via your SSH client.

To view the status:

sudo systemctl status code-server

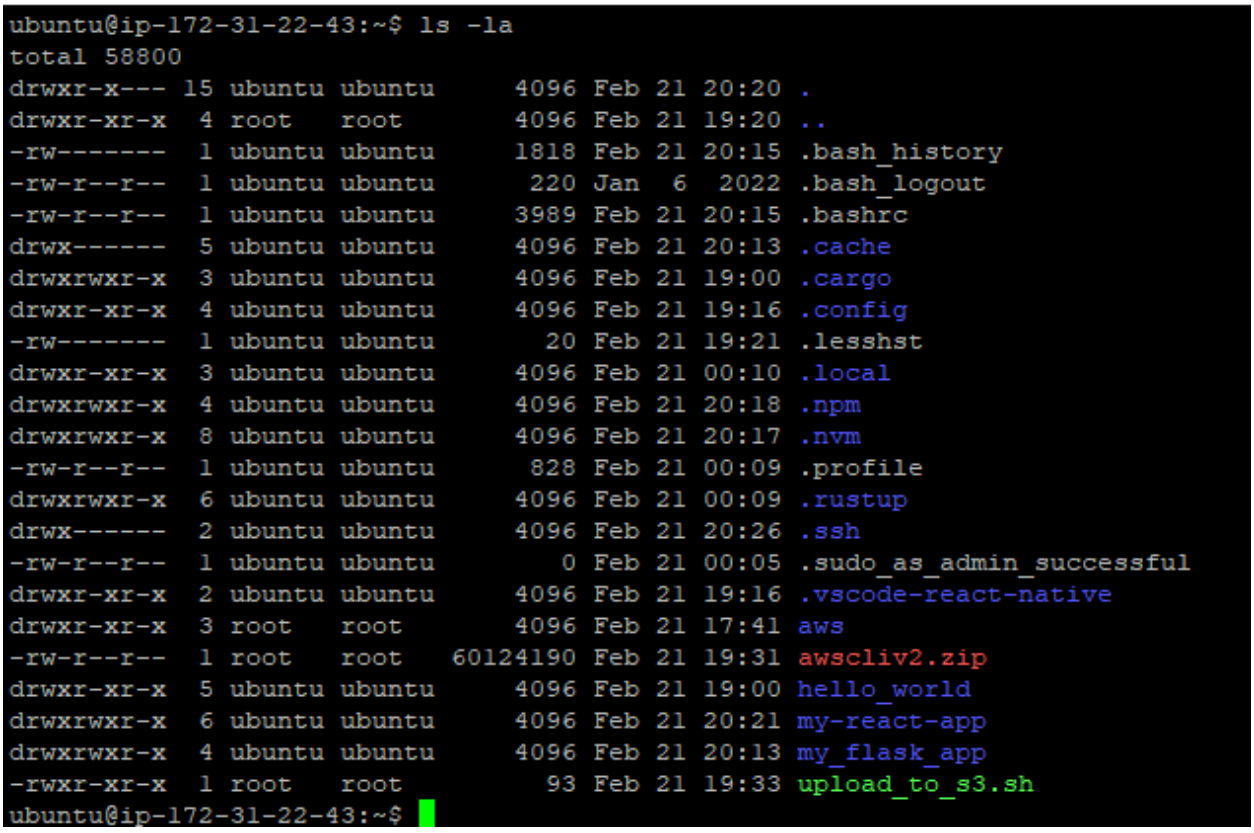
If you need to start or restart

sudo systemctl start code-server

sudo systemctl restart code-server

To update software, use: **sudo apt-get update**

At Ubuntu Prompt use: **ls -la** command to view the directories



For Flask Users:

1. Log into the Visual Code Server in web browser to set up a Virtual Environment for a Flask project: **ls -la**

2. Navigate to your project directory. At the Ubuntu prompt run:

   cd my_flask_app

   python3 -m venv venv

3. Activate the Virtual Environment:

   source venv/bin/activate

4. Edit app.py to contain a simple Flask application. For example: "Hello, World"

   sudo nano app.py

5. **Exit & Save**

6. Run Your Flask Application:

   flask run --host=0.0.0.0

7. In a browser view the output: **http://your_instance_public _ip:5000**

   - For ex: **http://32.05.62:5000**

For React Users:

1. Log into the Visual Code Server in web browser to set up a Virtual Environment for React App: **ls -la**

2. Navigate to your project directory. At the Ubuntu prompt run:

   cd my-react-app

3. Start the React Development Server:

   npm start

4. In a browser view the output: **http://your_instance_public _ip:3001**

   - For ex: **http://32.05.62:3001**

For Rust Users:

1. Log into the Visual Code Server in web browser to set up a Virtual Environment for Rust Project: **ls -la**

2. Navigate to your project directory. At the Ubuntu prompt run:

   cd hello_world

3. Build and Run Your Rust Project. Compile and run your new project with:

   cargo run

4. You should see output indicating that your Rust project has compiled and run, typically printing **"Hello, world!"** to the console.

OPTIONAL

Save files to S3 bucket
To save code or any files from your AWS EC2 instance directly to an Amazon S3 bucket you can use the AWS CLI.

Quick Guide Setup: **Replace the names below with your newly created items.**

1. Go to the AWS Console and select S3 Dashboard. Next select "Create a bucket".

2. Change the Object Ownership to: "**ACLs enabled**" and **grant** public access to this bucket by "**unchecking**" for Block all public access.

3. **Save changes.**

   In this example the bucket name is: myproject-bucket-aws

4. Inside that Bucket. " **Create folder**". In this example the folder is named: "Shatz"

5. Upload a file to this folder. In this example, the file is a text file (.txt file) named "Shatzt.txt"

6. Be sure under "**Permissions**" tab, that "Everyone (public access)" is checked for granting public access to this folder and file.

7. Next go back to your **AWS Console Home** and log into the to the **"IAM"** section.

8. Create an IAM role for your Instance.

   a. Go to: "**Roles**" under "**access management**"

   b. Click "**Create role**"

c. Select “**AWS service**” & “**EC2**” for the “use case”

d. Click “**Next**”

e. In search for permission policies, type “**AmazonS3FullAccess**’

f. Check “**AmazonS3FullAccess**” and click “**Next**”

g. Name the Role and “**click** “Create role”

7. Go back to your AWS console, select > **EC2 Dashboard** > **Instances running**.

a. Check your “Instance” that is running.

b. Under ‘**Actions**”, select “**Security**” and then “**Modify IAM Role**”

c. **Search** for the role you just created in the search tab and select it.

d. Then click “**Update IAM role**”

*To view your S3 buckets, at the ubuntu command prompt in Visual Studio, run: **aws s3 ls**

Download a file from S3 to Visual Code Studio Replace with your file’s names

1. Open your terminal on the Visual Studio Code Server.
2. Run the AWS CLI command to download the file. Choose a local path where you want to save the file. If you want to save it in your home directory, you can use the following command:

aws s3 cp s3://myproject-bucket-aws/Shatz/Shatzt.txt /home/ubuntu/Shatzt.txt

*This command downloads Shatzt.txt from your specified S3 bucket and saves it to **/home/ubuntu/Shatzt.txt**.*

Download a file from Visual Code Studio to S3 bucket

1. Access Visual Studio Code Server in your web browser and open a new terminal window. You can usually open a terminal via the Terminal menu at the top or by using a shortcut (often **`Ctrl+``**).

Navigate to the File's Location

2. Navigate to the directory containing **Shatzt.txt.** If you followed the previous instructions, it should be in /home/ubuntu/. You can ensure you're in the correct directory by running:

cd /home/ubuntu/

Upload the File to S3

3. Upload the file using the AWS CLI. Run the following command to upload Shatzt.txt back to the S3 bucket in the specified directory:

aws s3 cp Shatzt.txt s3://myproject-bucket-aws/Shatz/Shatzt.txt

This command specifies the local path to **Shatzt.txt (Shatzt.txt,** assuming you're already in the **/home/ubuntu** directory) and the S3 destination (**s3://myproject-bucket-aws/Shatz/Shatzt.txt**), effectively overwriting the previous version in the bucket with your updated file.

Additional Tips

- Verify Upload: To ensure your file was uploaded successfully, you can list the contents of the S3 directory again:

  aws s3 ls s3://myproject-bucket-aws/Shatz/

- Use Relative or Absolute Paths: Note that in the '**aws s3 cp'** command, you can use either relative or absolute paths for the local file. If you're not in the same directory as the file you wish to upload, you'll need to provide the absolute path or change your current directory to where the file is located.
- AWS CLI Configuration: Make sure the AWS CLI is configured with credentials that have sufficient permissions to write to the specified S3 bucket. If you haven't done this or if you encounter permission errors, you may need to run '**aws configure'** to set up your credentials.

By following these steps, you should be able to upload files from your Visual Studio Code Server environment to your AWS S3 bucket seamlessly.

AWS Data

- Data Encryption Configuration: This solution does not encrypt data within the running instance.
- User Credentials are stored: /root/.ssh/authorized_keys & /home/ubuntu/.ssh/authorized_keys
- Monitor the health:
    - Navigate to your Amazon EC2 console and verify that you're in the correct region.
    - Choose Instance and select your launched instance.
    - Select the server to display your metadata page and choose the Status checks tab at the bottom of the page to review if your status checks passed or failed.

Extra Information: (Optional)

Allocate Elastic IP

To ensure that your instance **keeps its IP during restarts** that might happen, configure an Elastic IP. From the EC2 console:

1. Select ELASTIC IPs.
2. Click on the ALLOCATE ELASTIC IP ADDRESS.
3. Select the default (Amazon pool of IPv4 addresses) and click on ALLOCATE.
4. From the ACTIONS pull down, select ASSOCIATE ELASTIC IP ADDRESS.
5. In the box that comes up, note down the Elastic IP Address, which will be needed when you configure your DNS.
6. In the search box under INSTANCE, click and find your INSTANCE ID and then click ASSOCIATE.
7. Your instance now has an elastic IP associated with it.
8. For additional help: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html