**AWS Solutions Architect Bootcamp**

**Week 1**

1. **Install Git Bash**

Visit the Official Git Bash Website and Run the Installer depending on the type of system you have using either of the links below:

* If you have a 32 bits laptop, download this git installer [https://github.com/git-for-windows/git/releases/download/v2.30.0.windows.2/Git-2.30.0.2-32-bit.exe](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://github.com/git-for-windows/git/releases/download/v2.30.0.windows.2/Git-2.30.0.2-32-bit.exe%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1614372394546000%26amp;usg%3DAOvVaw0ghXDO-e7DP4mJ7YnTaBS3&sa=D&source=editors&ust=1614372394631000&usg=AOvVaw1PSrctI-6kIo2E-NU5cPVd)
* If you have a 64 bits laptop, download this git installer  [https://github.com/git-for-windows/git/releases/download/v2.30.0.windows.2/Git-2.30.0.2-64-bit.exe](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://github.com/git-for-windows/git/releases/download/v2.30.0.windows.2/Git-2.30.0.2-64-bit.exe%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1614372394546000%26amp;usg%3DAOvVaw0HErqpgTPm4l4zGWi8gS5P&sa=D&source=editors&ust=1614372394631000&usg=AOvVaw035BHXM30xZCX6pgHs4bMn)

Validate all the configurations until you are able to run the installation.

Launch the Git bash and Happy Coding!

1. **Create an Account with Git Radar**

Use the link below to sign up for Git Radar by just entering your username and your password: [https://gitradar.com/signup](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://gitradar.com/signup%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1614372394548000%26amp;usg%3DAOvVaw0neQ84qjAm-AyzsIx28JVJ&sa=D&source=editors&ust=1614372394631000&usg=AOvVaw1NIwkDaIfMslyAyKG4A6ts)

**Steps to Cloning a Repo from Git Radar**

* **Step 1:**Create an empty repo on Git radar by entering the repo name and clicking on submit. Make sure repo name has no spaces or special characters such as @, #, &...
* **Step 2:**If this is your first time using Git Radar, you have to Activate your SSH (Secure SHell) key using the following steps else skip to **Step 3**:
* In your Git bash terminal, type the command ***ssh-keygen -t rsa***. This command generates two files, one containing your public key and another containing your private key. The public key helps your local computer(Client Side) connect to other networking systems (Server Side) such as Git radar in our case. Never share your private key with anybody.

Press enter until your key image is being generated

* In the output you get from running the previous command in the terminal, carefully copy the path in which your public key was being saved without including any whitespaces. Example of path, */c/Users/Your\_PC\_Name/.ssh/id\_rsa.pub*
* Then run the command ‘cat path\_you\_copied’. The cat command displays the content of a file path, in our case, it displays our public key, which is a random combination of letters and numbers unique only to our local computers.

Example, ***cat /c/Users/Your\_PC\_Name/.ssh/id\_rsa.pub***

* Copy this key without including any extra space
* Go to your profile on Git Radar, at the top right of the page,  click on the drop down menu and then click on *Manage SSH keys*
* Click on upload a key, then paste your key in the space provided and submit.
* You have successfully activated your SSH key.
* **Step 3**: Click on the copy icon next to the repo name you wish to clone. Clicking will copy a path specific to that repo.
* **Step 4**: Back in your Git bash terminal, type the following command, git clone + paste the path you copied. Avoid using *ctrl + c* or *ctrl + v* on the Git Bash terminal, it’s going to give you unexpected results and not the copy and paste you were expecting. Change the numbers in the path you copied to 51000. This can be achieved with the help of the left arrow key. Example,

***git clone ssh://***[***user@gitradar.com***](mailto:user@gitradar.com)***:51000/home/user/git/repo\_name.git***

* **Step 5**: You have successfully cloned your repo. You can now navigate into the folder using the ***cd****(see Glossary)* command and start making changes to build your application.

1. **Create, Modify & Add Data to a File using Git**
2. ***touch filename:***In the Git Bash terminal, run this command to create a new file with any name of your choice e.g *touch mynewfile.txt*

Make sure you are in the repo folder you cloned. You can check this using the ***pwd****(see Glossary)*command.

1. Once your file has been created, you can go to where the file is located in your file explorer, open this file and add whatever you want to write in this file. Save your changes and close the file.

        Now we have our changes saved on our local computer, (Client S

ide). To save these changes to Git Radar, (Server Side), we will be using a couple of git commands to do this. See Glossary for what each command does.

1. ***git status:***This command is just to check the status of your repo. Whether or not there are changes to be committed.
2. ***git add .***
3. ***git commit -m “****write your commit message here****”***
4. If this is your first time using the Git bash terminal, you have to configure your email address and your name. But if not skip to the next commands.

* ***git config --global user.name <your name here>***
* ***git config --global user.email <your email here>***
* Then run the above git commit command again, ***git commit -m “****write your commit message here****”***

1. ***git log***
2. ***git push***

***Glossary***

**Git Commands**

|  |  |
| --- | --- |
| git clone <repo> | Clones or copies a repo located at <repo> into your local machine. |
| git add . | Adds or Stages all changes made for the next commit. |
| git commit -m "<message>" | Commits the staged files, use <message> as the commit message that explains the changes you made. |
| git status | List which files are staged, unstaged, and untracked. Tells you the state of your files, whether they need to be added, committed or pushed. |
| git log | Displays the recent commit history using the default format. |
| git log --all | Displays the entire commit history using the default format. |
| git push | Push all of your local changes to the server side you are using e.g Git Radar |
| git pull | Pull all changes from the Server side to you local computer (Client side) |
| git branch | Check the branches in your Repo |
| git checkout <branch name> | Used to switch between different branches |
| git config --global user.name <name> | Define the author name to be used for all commits by the current user. |
| git config --global user.email <email> | Define the author email to be used for all commits by the current user. |

**Other useful commands**

|  |  |
| --- | --- |
| pwd (present working directory) | Shows you the current directory or folder path in which you are working in |
| cat <file path> | Displays the content of the file in the file path |
| ls -l | List all folders or files in the current directory |
| ls -la | List all folders or files in the current directory including hidden files. Hidden files are files beginning with a full stop e.g *.sshfile* |
| ls -ltr | List all folders and files sorted according to time created |
| cd <filename> | Changes directory to the filename |
| cd ~ | Changes your directory and takes you to your home directory |
| touch <filename> | Creates a file called filename |
| rm <filename> | Removes a file called filename |

**VIM Commands**

|  |  |
| --- | --- |
| vim <filename> *or* vi <filename> | Creates a new file called filename if this doesn’t already exist, but if it exists then it just opens the file |
| i (insert) | When in vim mode, the i command enables you to insert or modify the current file |
| escape key + :wq | Exits and Saves the changes made to your file |
| escape key + :q! | Exits and closes the file without saving the changes made |
| escape key + u | Undo changes made |

**Week 2**

1. **Develop First Web Page using HTML**

HTML, HyperText Markup Language, gives content (web pages) structure and meaning by defining that content as, for example, headings, paragraphs, or images. CSS, Cascading Style Sheets, is a presentation language created to style the appearance of your content using, for example, fonts or colors. In summary, HTML will always represent content, and CSS will always represent the appearance of that content. Follow the steps below to create a simple web page using an html file,

* In your cloned Repo on Git bash, create a new file named index.html using the touch command, ***touch index.html***
* Open the index.html file to edit it on VIM using the command ***vim index.html***
* Press **i** on the keyboard to enable editing or insert mode on the open empty html file on VIM. You should see **-- INSERT --**below the screen to make sure you are able to edit your file
* Go ahead and paste the following html code inside your file, by right clicking and click on paste. Keyboard shortcuts like *ctrl + v* doesn't work on VIM,

<!DOCTYPE html>

<html>

 <head>

   <meta charset="utf-8">

   <title>Welcome to Datagend Bootcamp</title>

 </head>

 <body>

   <p>Hello, I am Your\_Name\_Here and I am a Computer Science rockstar.</p>

 </body>

</html>

* Save your changes made using ESC and then typing out ***:wq***
* You can now test your new web page by going to the directory containing your index.html file. Double click on the index.html file or right click and open with chrome.
* And now you should be able to see your web page in the browser displaying the text, *“Hello, I am Your\_Name\_Here and I am a Computer Science rockstar.”*

You can make your webpage even more user friendly by adding an image to the page.

* Copy an image to the folder containing the index.html file
* On Git bash, use the ls command to make sure your image file is in the same directory as your index.html file. Copy the name of your image that can be *picture.jpg*for example.
* So let’s open up our index.html file again on VIM and go to insert mode as above using ***vim index.html***and ***i***.
* Using the arrow keys to move around inside your file, just below the opening body tag, paste the following html code,

<img src="image\_name" alt="Rockstar" width="200" height="200">

Where image\_name is the name of the image file you copied from above e.g *picture.jpg.*Your code should now look like this,

<!DOCTYPE html>

<html>

 <head>

   <meta charset="utf-8">

   <title>Welcome to Datagend Bootcamp</title>

 </head>

 <body>

        <img src="image\_name" alt="Rockstar" width="200" height="200">

   <p>Hello, I am Your\_Name\_Here and I am a Computer Science rockstar.</p>

 </body>

</html>

* Save your changes using ESC and :wq
* Go back to the file explorer where the index file is located and test your webpage by opening it. You should see your beautiful image and a paragraph on the web page.
* You can now save your changes on Git Radar using ***git add .***, ***git commit -m “your message”*** and ***git push***

Congratulations, you have just built a web page. But what good is a web page if you can not share with your friends to see what you built? To achieve this, we will be deploying our code on Git Radar.

1. **Deploy Web Page on Git Radar**

* On git radar, click on the arrow or plane-like icon on your repo
* Add the branch name of your repo to the form that appears. You can check your branch name on git bash using, ***git branch***
* Type in a subdomain name and click on deploy
* You can click on the arrow icon again, scroll up and click on *view deployments.*Copy your subdomain name and paste it on a new browser and your code should be live on the internet.
* Go ahead and share your subdomain link with your friends to see your web page.

        More useful resources to study HTML & CSS, [w3school online web tutorials](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://www.w3schools.com/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1614372394586000%26amp;usg%3DAOvVaw2xrM_V7Fs4jqxk9ZI8yaxO&sa=D&source=editors&ust=1614372394638000&usg=AOvVaw10yLrMYiHk2NJfJHaBoYav) and [building your first web page](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://learn.shayhowe.com/html-css/building-your-first-web-page/%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1614372394586000%26amp;usg%3DAOvVaw3fjraYi-_OB27ojAqrCcqM&sa=D&source=editors&ust=1614372394638000&usg=AOvVaw1nulhDzcgN9w7mHm2d56m9).

1. **Introduction to Terraform**
2. **Introduction to Infrastructure as Code**

* Terraform is the infrastructure as code tool from HashiCorp.
* It is a tool for building, changing, and managing infrastructure in a safe, repeatable way.
* Operators and Infrastructure teams can use Terraform to manage environments with a configuration language called the HashiCorp Configuration Language (HCL) for human-readable, automated deployments.
* If you are new to infrastructure as code as a concept, it is the process of managing infrastructure in a file or files rather than manually configuring resources in a user interface. A resource in this instance is any piece of infrastructure in a given environment, such as a virtual machine, security group, network interface, etc.
* At a high level, Terraform allows operators to use HCL to author files containing definitions of their desired resources on almost any provider (AWS, GCP, GitHub, Docker, etc) and automates the creation of those resources at the time of application.
* While many of the current tools for infrastructure as code may work in your environment, Terraform aims to have a few advantages for operators and organizations of any size.
* Platform Agnostic
* State Management
* Operator Confidence

1. **Install Terraform**
2. On a Windows machine. Download from Terraform from the website here: [https://www.terraform.io/downloads.html](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://www.terraform.io/downloads.html%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1614372394590000%26amp;usg%3DAOvVaw31jIH_MawbQlAP5JOsQjFQ&sa=D&source=editors&ust=1614372394639000&usg=AOvVaw2cZ3XJJnFEe22R5NBxkVNL)and place it in any folder like “downloads”.
3. Extract/Unzip the binary files downloaded above
4. Once the pop up box displays, replace what's there with c:\terraform (type over it).
5. Add the binary to the path on windows. To do this,
6. type “system” in the search window.
7. in the pop up that has (control panel>system and security>system, choose “Advanced system settings” in the left pane.
8. Click Environment Variables and in the new pop up box, select path in the System variables section, and click “Edit”
9. Click New and add C:\Terraform
10. Go to the Search box in the windows box at the bottom left and type cmd to see if we did install correctly in the command line.

At the prompt, type terraform --version. You should see the version you installed.

1. **Build Infrastructure**

**Week 3**

1. **Add CSS to your Web Page**
2. **Add JS to your Web Page**
3. **How to Access your Remote Linux Computer(Git Radar Workspace)**

* Open your git bash command and make sure you are in the home directory. You can get there using **cd ~**
* Now type in the following command on your git bash terminal.

**ssh**[**username@gitradar.com**](mailto:username@gitradar.com)**-p 51000**

Make sure to replace the username in the above command with your git radar username, and remember that it is case sensitive so make sure to type it in exactly the way you created the username.

* Type in **exit** to exit your remote computer

1. **AWS CLI**
2. **Install AWS CLI**

* For windows download the following zip file, unzip it and run it for installations:

[https://awscli.amazonaws.com/AWSCLIV2.msi](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://awscli.amazonaws.com/AWSCLIV2.msi%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1614372394593000%26amp;usg%3DAOvVaw0o6uivUNzCSxzdFPcJuzas&sa=D&source=editors&ust=1614372394640000&usg=AOvVaw0ypbj4J_sEt509Mvca-TMZ)

* For linux, run the following commands:
* *cd ~*
* *curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"*
* *unzip awscliv2.zip*(if you don’t have this command you can install it using, *sudo apt install unzip*, then run the command on this step again)
* *sudo ./aws/install*
* *aws --version*Run this command to check if you have successfully installed aws cli. You should have the following output: *aws-cli/2.1.22 Python/3.7.4 Linux/4.14.133-113.105.amzn2.x86\_64 botocore/2.0.0*

1. **Configure your AWS CLI Profile**

* Make sure you have an AWS user with administrative access.
* In your git bash, run the  command  *aws configure --profile <yourgitradar username*> and fill in the following information: Access key ID, Secret key (both from the aws management console), Region name (use us-east-1), Output format (json).
* To confirm that you successfully configured it, run the following commands
* cd ~/.aws
* cat config
* cat credentials

                        And you should see all the information you entered earlier.

**Week 4**

**What is Serverless**

Serverless architecture (also known as serverless computing or function as a service, FaaS) is a software design pattern where applications are hosted by a third-party service, eliminating the need for server software and hardware management by the developer. A **serverless** architecture is a way to build and run applications and services without having to manage infrastructure. Your application still runs on servers, but all the server management is done by AWS, Google Cloud, Azure or any cloud provider you choose to work with.

**Install Serverless using NPM**

We need to have node js and npm install so we can use it to install serverless.

* Download the latest version of node using the following link, [https://www.npmjs.com/get-npm](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://www.npmjs.com/get-npm%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1614372394598000%26amp;usg%3DAOvVaw2D1y24c2Pk0r3H0OmSwWSa&sa=D&source=editors&ust=1614372394641000&usg=AOvVaw0_WM7eKwvgaB9y8tXfc7A6)

To make sure you have them installed, check their versions as follows;

*node -v*

*npm -v*

If you successfully downloaded node and npm, you will see the versions displayed on the terminal.

* Now we can install serverless using npm as follows:

npm install -g serverless

Run serverless  to make sure it was installed globally.

* Now you can configure your serverless using:

serverless config credentials --provider aws --key AWSAccessKeyID --secret AWSSecretKey --profile username

Replace AWSAccessKeyID and AWSSecretKey with your actual IAM user credentials.

* Create a template now using AWS or any cloud provider of your choice.

serverless create --template aws-nodejs --path YourPath

Note that a folder will be created called YourPath  so you can cd to cd YourPath then launch your VS code editor using code .

Don’t forget the dot after the previous code command. This code will work only if you have VS Code editor installed.

You can install it from this site [https://code.visualstudio.com/download](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://code.visualstudio.com/download%26amp;sa%3DD%26amp;source%3Deditors%26amp;ust%3D1614372394600000%26amp;usg%3DAOvVaw1NfEp-qr3x7D_zKvIwPHwA&sa=D&source=editors&ust=1614372394642000&usg=AOvVaw0vPZqnOCBw2YV7ICMZSA1c) depending on your OS.

* Run serverless package which will package all your files into template files.
* To deploy your template make sure you are in the YourPath  directory. Then run serverless deploy
* To test your lambda function on the terminal, run serverless invoke --function functionName
* You can remove the function using serverless remove