Starting from Scratch: Manual AWS Infrastructure

Creation of AWS infrastructure manually to host our application is performed in this lesson. We then will install our web application on an EC2 instance.

We'll cover the following

Objective

- Get a simple web application running on a single EC2 instance.

Steps

- Manually create basic AWS infrastructure to host our application.
- Manually install our application on an EC2 instance.

Hosting our application

If you don't already have an AWS account, you will need to create one.

> ⚠ You should never use your AWS root account credentials other than to create an Administrator user for your account. AWS has several best practices for managing accounts and credentials.
>
> If you are creating a new AWS account or you don't yet have an Administrator user, use the root account to create an Administrator user now, and then use only that user.

> 🔍 We've chosen to provide links to the AWS console in the US East (N.

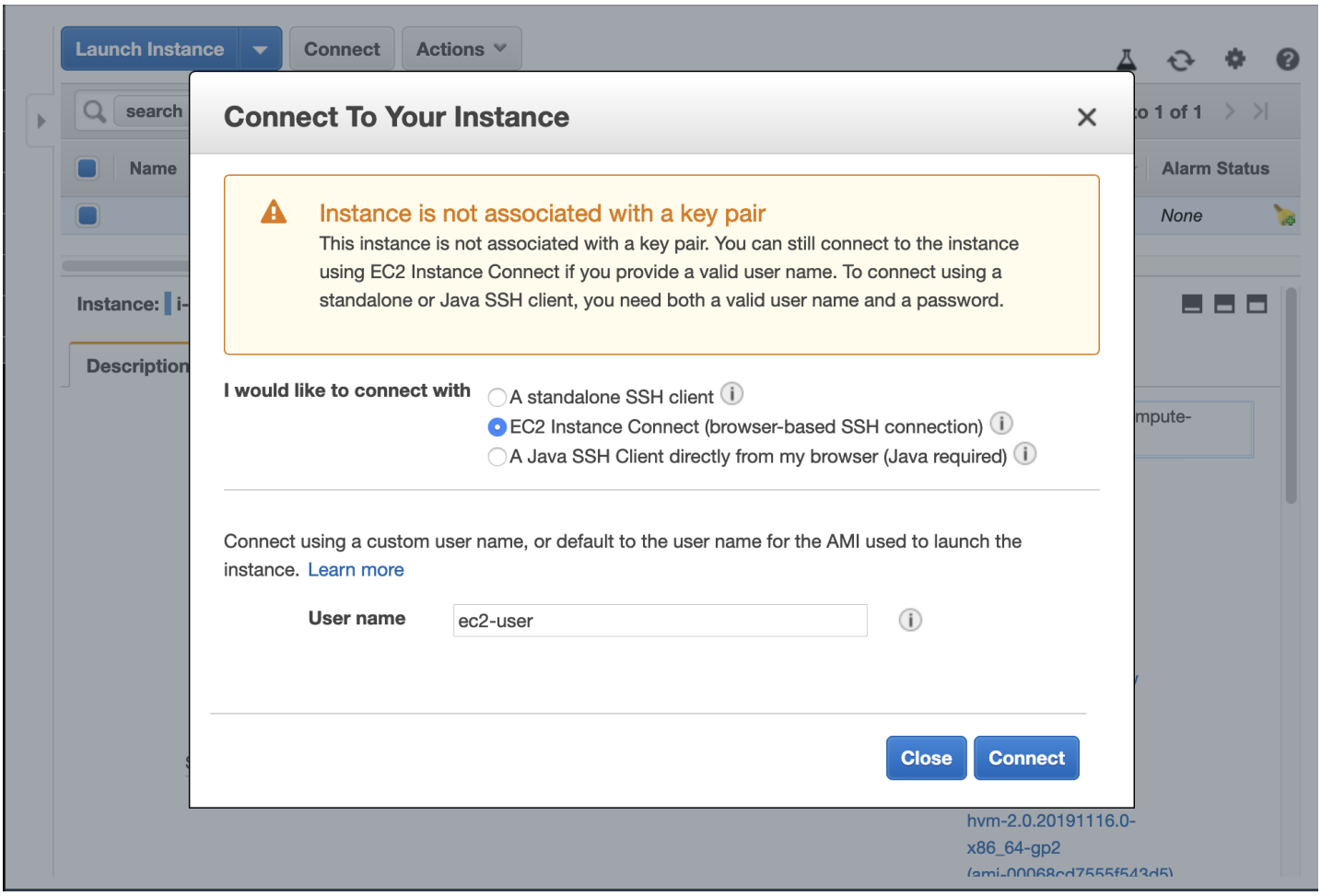
> Virginia) (us-east-1) region. Feel free to choose a region closer to your physical location. Everything described here is available in all AWS regions.

Now we're going to use the AWS console to create the minimal resources necessary to get our application running in our AWS account.

Note the Public DNS (IPv4) field shown in the console. This address will allow us to reach our instance via the internet.

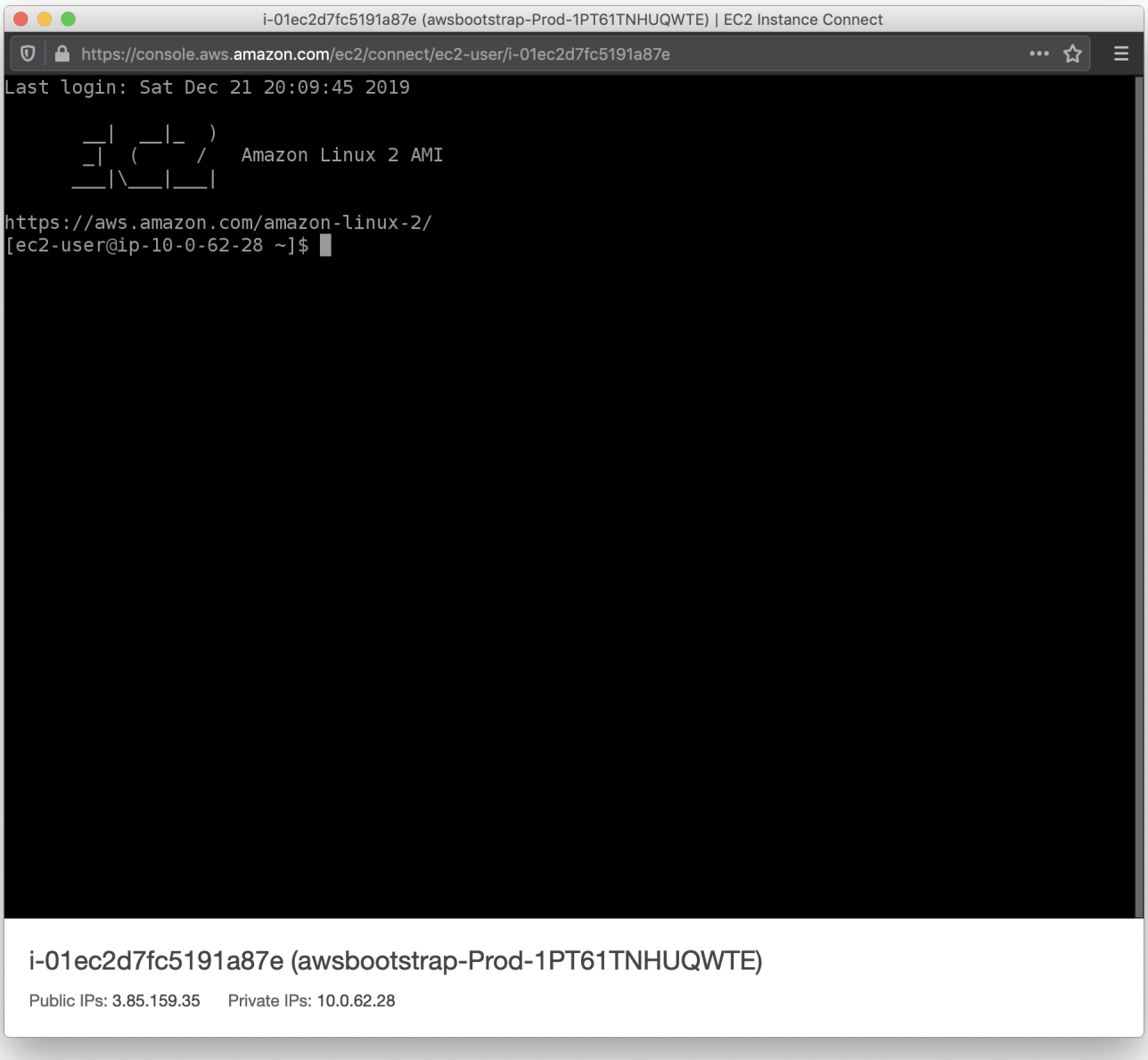
Running our application

For this exercise, we're going to SSH to our instance from the browser. In the EC2 instances view, select our instance and then press the *Connect* button at the top of the page to start the connection. Then select *EC2 Instance Connect* and hit *Connect* at the bottom of the dialogue.



EC2 Connect

A new browser window will pop up with an open SSH connection to our instance.



EC2 Connect SSH Connection

Now that we have an SSH shell open on the host, we're going to update the installed packages manually (for now). Then we will install our application and its dependencies.

```
sudo yum -y update
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash
. ~/.nvm/nvm.sh
nvm install node
```

terminal

Line #1: Updates the installed yum packages.

Line #2: We'll use NVM to install node

Line #2: We'll use NVM to install node.

Line #3: Makes sure NVM is available.

Line #4: Installs node via NVM.

Now that our dependencies are in place, we can proceed to install our application.

```
mkdir logs
curl -sL https://github.com/<username>/aws-bootstrap/archive/master.zip --output master.zip
unzip master.zip
mv aws-bootstrap-master app
cd app
npm install
npm start
curl localhost:8080
Hello World
```

terminal

Line #2: Replace `<username>` with your GitHub user name.

Line #2: Here we use GitHub's archive to download the latest version of our application. This works only with public repositories.

At this point, if you copy the Public DNS (IPv4) from *Figure 6* (See the *sixth* slide in the above slideshow), you should be able to point your web browser to `http://<public_dns>:8080` and see the "Hello World" message from our application.

Congratulations! We now have our application running in the cloud. However, our infrastructure is not yet fault-tolerant or scalable. And it's not particularly easy to configure again, either. But we'll get there, step by step, in the following sections.

> At this point, you may also want to set up billing alerts with CloudWatch to help you monitor your AWS charges and to remind you if you forget to decommission any experiments you have performed using your AWS account.

In the next lesson, we will recreate our infrastructure using CloudFormation.