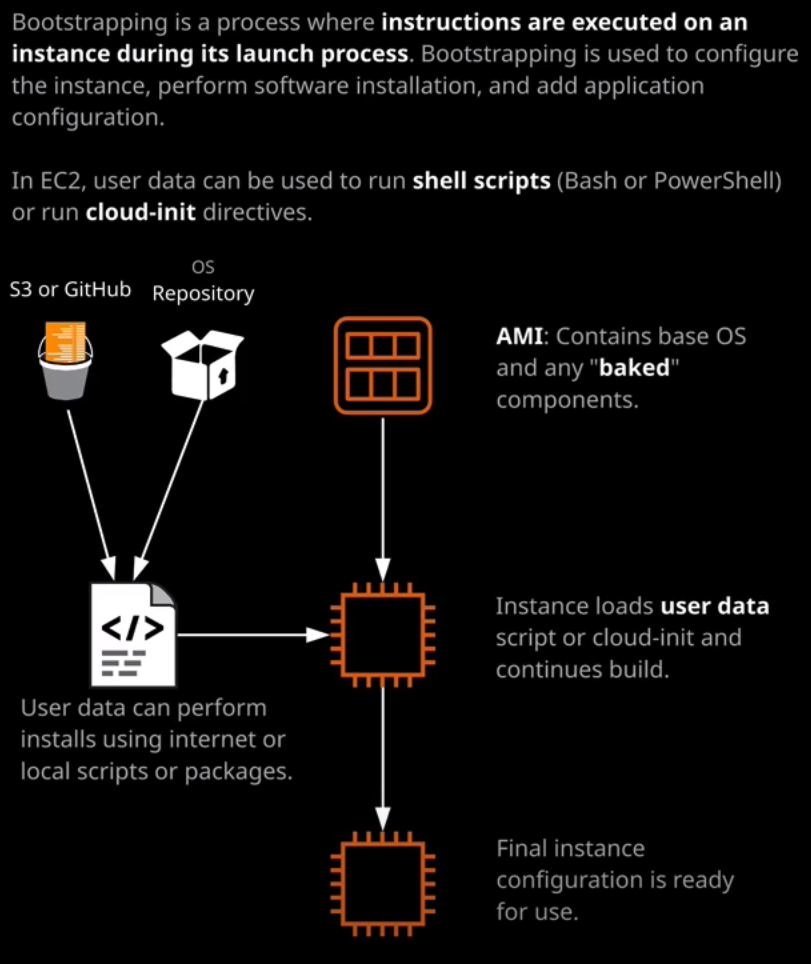
**Bootstrapping is the process of giving an instance a series of commands to do as part of its provisioning process.**

Bootstrapping is what allows us to customize what the instance will do and how it's configured. It means that by the time an instance is provisioned, it's ready to go, configured in a certain way, with certain capabilities



**User data** is how EC2 performs bootstrapping.

User data is the EC2 implementation of bootstrapping. Essentially, whatever you paste in this user data section is used by the EC2 instance, when it's first provisioning so it can install applications, it can perform certain configurations, it can pull down data from a GitHub repository or an S3 bucket. Anything that you can put as a script or a script like set of directives could be given inside this user data.

Now this user data can take a number of different forms. It can either be a shell script or it can use something called cloud init. If it's a shell script that takes the form of either a PowerShell script if it's Windows or a Bash script if it's Linux. In the previous lesson, this is the Bash script that we used to install and set up the Cat Hall of Fame website so this has been adjusted slightly to be this lesson's script.   
  
Shell Script:

#!/bin/bash **//we are indicating that we want to use bash**yum update -y **// yum is the package manager to install the system update**  
yum install -y httpd **//install web server**yum install -y wget **// install utility called wget**chkconfig httpd on   
cd /var/www/html   
wget https://raw.githubusercontent.com/linuxacademy/content-aws-csa2019/master/lesson\_files/03\_compute/Topic2\_Intermediate/02\_Bootstrapping/index.html   
wget https://raw.githubusercontent.com/linuxacademy/content-aws-csa2019/master/lesson\_files/03\_compute/Topic2\_Intermediate/02\_Bootstrapping/catanimated.gif   
wget https://raw.githubusercontent.com/linuxacademy/content-aws-csa2019/master/lesson\_files/03\_compute/Topic2\_Intermediate/02\_Bootstrapping/rainbow.gif   
wget https://raw.githubusercontent.com/linuxacademy/content-aws-csa2019/master/lesson\_files/03\_compute/Topic2\_Intermediate/02\_Bootstrapping/penny.jpeg   
wget https://raw.githubusercontent.com/linuxacademy/content-aws-csa2019/master/lesson\_files/03\_compute/Topic2\_Intermediate/02\_Bootstrapping/roffle.jpeg   
wget https://raw.githubusercontent.com/linuxacademy/content-aws-csa2019/master/lesson\_files/03\_compute/Topic2\_Intermediate/02\_Bootstrapping/truffs.jpeg   
wget https://raw.githubusercontent.com/linuxacademy/content-aws-csa2019/master/lesson\_files/03\_compute/Topic2\_Intermediate/02\_Bootstrapping/winkie.jpeg   
service httpd start **//making sure httpd the web service is started and ready to use the assets**

EC2 is capable of using something slightly different that's known as cloud init. Now, rather than a series of direct operating system instructions, cloud init enables you to provide directives. Now they can be operating system explicit command. So in this case we're still using wget to download the assets from the GitHub repository, but they can also be desired state commands. So these are operating system agnostic commands that essentially reflect what you want the instance to do. If we're using cloud init, we can abstract away from the actual operating system. Using cloud init it is a little bit more complicated, but it is really powerful if you're using this in production.

Now, the main takeaway that I want you to have from this lesson is why you'd use bootstrapping versus AMIs.

**The biggest advantage to baking any configuration into an AMI is to reduce the time it takes to provision that instance**. So if you got any applications that take a long period of time to install and configure then they're candidates for **baking into the AMI**. An example of this is if you need to get new instances into production in less than 20 minutes and an application takes 45 minutes to install and configure, then you know you can't use bootstrapping. Bootstrapping in that scenario would be including the shell commands in the user data that would perform the installation of the application automatically but if it takes 45 minutes, it's going to be 45 minutes before your instance is ready for production. So in that case, you'd use an AMI. You get a source instance, you perform the installation, get everything ready to go, and then create an AMI from that source instance, and that would mean you could bring your instance into production readiness within a couple of minutes. **The downsides to using AMI is that you don't really have the ability to do any dynamic configuration.** So if you bake everything into the AMI, it's already going to be configured. You can't really do any configuration based on the size of the instance, the type of the instance, the subnet the instance is in, or any DNS or IP addressing. Anything specific that you want to be dynamic if you use 100% AMI baking you don't have that option. It's all done.

Now choosing between AMIs and bootstrapping doesn't have to be a binary one or the other decision you can choose to use and AMI to bake in lot of the really intensive parts of building an application. So if you're doing an installation and that installation say takes 20 or 30 minutes and then after the installation, you can do some configuration. **What you might want to do is AMI bake the installation and then perform the configuration dynamically, using bootstrapping.** For the exam, remember, it's not a one or the other. You've got the capability to **AMI bake some stuff and you want to do that when it's lengthy, when it's time intensive to do, and when it could impact the ability to bring things into production. Whenever you've got anything that's dynamic when you want to do it at instance provisioning time that's when you'd used bootstrapping** and generally, most production systems use a combination of the two. There are exam questions on this, so when you do face an exam question that might mention AMI baking or bootstrapping, remember, **AMIs are for your lengthy static configurat**ion. **Bootstrapping is what you use for anything dynamic**. When you're facing a question that talks about, you need to get an instance into production within a certain time frame and then it says that the application might take 20 or 30 minutes to install focus on the timeframes. It's those that will give you the clues whether you need to use AMI baking or bootstrapping. If you need to do the work upfront. ***If you need to save time, bake it into the AMI. If you need to do any dynamic configuration then use*** ***bootstrapping.***