aws ubuntu download file from bitbucket

Continuous Deployment Pipeline with Bitbucket-Pipelines to AWS EC2 using AWS Code Deploy.

Step-by-step journal with best AWS practice, Bitbucket Pipelines and more …

Just in case, you'd like to explore on your own, you may find project repo here .

To start with, let's introduce the tool and the platform we may apply in this project.

Bitbucket Pipelines is an integrated CI/CD service, built into Bitbucket. It allows you to automatically build, test and even deploy your code, based on a configuration file in your repository. Essentially, we create containers in the cloud for you. Inside these containers you can run commands (like you might on a local machine) but with all the advantages of a fresh system, custom configured for your needs.

Amazon Web Services (AWS) is a secure cloud services platform, offering compute power, database storage, content delivery and other functionality to help businesses scale and grow.

Now, let's dive deep into this project. What is the philosophy behind CD (continuous development)? Continuous Deployment (CD) is a software release process that uses automated testing to validate if changes to a codebase are correct and stable for immediate autonomous deployment to a production environment. The software release cycle has evolved over time.

Here is what we will accomplish at the end of this project : Deploying a simple Node.js Application on an AWS EC2 instance and automate its delivery every time a git push is initiated to your production repository. The code will be automatically deployed and hosted on your server with ease — click of a button.

Though this project only presents "Hello World" Node.js application but the concepts and methodologies being applied may utilize in other projects. You can definitely scale it based on your specific requirements and needs.

Here is how the app.js file for our Node server.

The directory structure looks like:

Now, let's get our hands dirty!

Overview:

Bitbucket repositories have pipeline support and we can use them to ease our deployments, you can read more about them here. They provide integrations for various platforms such as Amazon Web Services, Google Cloud Platform, Microsoft Azure, and many more.

We will be setting up an IAM user with programmatic access which will be used by bitbucket-pipelines to push a copy of our application to an S3 bucket and then trigger a deployment on AWS Codedeploy which will, in turn, deploy our application as per our appspec.yml configuration on our EC2 instance on our behalf.

In this project, I will provide you with step-by-step instructions to complete this project. Even though you are a total novice, no worries, you're at the right spot! We will make it together!

Step 1: Creating user and configure AWS.

Based on AWS best practice, root user should not be used to perform any task. So that we need to login to root user and create a user with suitable policies.

Ubuntu on AWS.

Whether you are moving to Amazon Web Services or are already running cloud-native, Ubuntu is the platform of choice for AWS.

Canonical continuously tracks and delivers updates to Ubuntu images to ensure security and stability are built-in from the moment your machines and containers launch.

Today, Ubuntu powers billions of mission-critical workloads.

UBUNTU ON AWS POWERS CLOUD LEADERS LIKE.

Choose the right Ubuntu for you.

Select between the standard Ubuntu Server and the Pro edition, which includes expanded security, compliance and AWS integration:

Ubuntu.

AWS-optimised kernel and userspace components built from the latest releases Broad set of open source application components available for development Security patches for the kernel and key infrastructure components Updates mirrored and images published in all AWS regions worldwide 5-year lifetime.

Ubuntu Pro for AWS.

Starting at 0.1c/hour.

Includes all Ubuntu optimisations and updates of the free version Expanded security covering application components delivered with Ubuntu Kernel live patching for instant security and longer uptimes FIPS 140-2 and CC-EAL certified components Integration with AWS security and compliance features 10-year lifetime.

Features.

Optimised.

Ubuntu on AWS runs on an AWS-optimised kernel, which includes improved device drivers, like ENA, and out of the box support for accelerators like GPUs. This means faster instance starts and better runtime performance for your workloads.

Security built in.

Our security teams track alerts 24x7 and release patches to system components and applications continuously. Canonical operates a worldwide distribution network ensuring that package updates are delivered in-region quickly.

How to use a Bash script to manage downloading and viewing files from an AWS S3 bucket.

As you can read in this article, I recently had some trouble with my email server and decided to outsource email administration to Amazon's Simple Email Service (SES).

The problem with that solution was that I had SES save new messages to an S3 bucket, and using the AWS Management Console to read files within S3 buckets gets stale really fast.

So I decided to write a Bash script to automate the process of downloading, properly storing, and viewing new messages.

While I wrote this script for use on my Ubuntu Linux desktop, it wouldn't require too much fiddling to make it work on a macOS or Windows 10 system through Windows SubSystem for Linux.

Here's the complete script all in one piece. After you take a few moments to look it over, I'll walk you through it one step at a time.

The complete Bash script.

We'll begin with the single command to download any messages currently residing in my S3 bucket (by the way, I've changed the names of the bucket and other filesystem and authentication details to protect my privacy).

Of course, this will only work if you've already installed and configured the AWS CLI for your local system. Now's the time to do that if you haven't already.

The cp command stands for "copy," --recursive tells the CLI to apply the operation even to multiple objects, s3://bucket-name points to my bucket (your bucket name will obviously be different), the /home/david. line is the absolute filesystem address to which I'd like the messages copied, and the --profile argument tells the CLI which of my multiple AWS accounts I'm referring to.

The next section sets two variables that will make it much easier for me to specify filesystem locations through the rest of the script.

Note how the value of the tmp_file_location variable ends with an asterisk. That's because I want to refer to the files within that directory, rather than the directory itself.

I'll create a new permanent directory within the . /emails/ hierarchy to make it easier for me to find messages later. The name of this new directory will be the current date.

I first create a new shell variable named today that will be populated by the output of the date +"%m_%d_%Y" command. date itself outputs the full date/timestamp, but what follows ("%m_%d_%Y") edits that output to a simpler and more readable format.

I then test for the existence of a directly using that name - which would indicate that I've already received emails on that day and, therefore, there's no need to recreate the directory. If such a directory does not exist (||), then mkdir will create it for me. If you don't run this test, your command could return annoying error messages.

Since Amazon SES gives ugly and unreadable names to each of the messages it drops into my S3 bucket, I'll now dynamically rename them while, at the same time, moving them over to their new home (in the dated directory I just created).

The for. do. done loop will read each of the files in the directory represented by the $tmp_file_location variable and then move it to the directory I just created (represented by the $base_location variable in addition to the current value of $ today).

As part of the same operation, I'll give it its new name, the string " email " followed by a random number generated by the rand command. You may need to install a random number generator: that'll be apt install rand on Ubuntu.

An earlier version of the script created names differentiated by shorter, sequential numbers that were incremented using a count=1. count=$((count+1)) logic within the for loop. That worked fine as long as I didn't happen to receive more than one batch of messages on the same day. If I did, then the new messages would overwrite older files in that day's directory.

I guess it's mathematically possible that my rand command could assign overlapping numbers to two files but, given that the default range rand uses is between 1 and 32,576, that's a risk I'm willing to take.

At this point, there should be files in the new directory with names like email3039, email25343, etc. for each of the new messages I was sent.

Running the tree command on my own system shows me that five messages were saved to my 02_27_2020 directory, and one more to 02_28_2020 (these files were generated using the older version of my script, so they're numbered sequentially).

There are currently no files in tmpemails - that's because the mv command moves files to their new location, leaving nothing behind.

The final section of the script opens each new message in my favorite desktop text editor (Gedit). It uses a similar for. do. done loop, this time reading the names of each file in the new directory (referenced using the " today " command) and then opening the file in Gedit. Note the asterisk I added to the end of the directory location.

There's still one more thing to do. If I don't clean out my S3 bucket, it'll download all the accumulated messages each time I run the script. That'll make it progressively harder to manage.

So, after successfully downloading my new messages, I run this short script to delete all the files in the bucket:

I'm an AWS solutions architect, Linux server professional, and author of books and Pluralsight courses on Linux, AWS, Docker, and IT security.

If you read this far, tweet to the author to show them you care. Tweet a thanks.

Learn to code for free. freeCodeCamp's open source curriculum has helped more than 40,000 people get jobs as developers. Get started.

freeCodeCamp is a donor-supported tax-exempt 501(c)(3) nonprofit organization (United States Federal Tax Identification Number: 82-0779546)

Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world.

Donations to freeCodeCamp go toward our education initiatives and help pay for servers, services, and staff.

Aws ubuntu download file from bitbucket.

Auto Code Deployment from BitBucket Pipelines to AWS EC2.

Purpose of this readme is to shorten the lengthy task of code committing, testing, building and then deploying on to the server.

There is actually an easier way to do that all. Won't it be nice if one git push and done, code is committed to the remote repository as well as built and deployed to your aws ec2 account?

Following steps might help you to do just that.

Following are some prerequisites before we can start.

Note : This demo is meant for BitBucket ONLY and not for gitlab or github.

AWS Account (For this demo purpose I am gonna use a free aws account which you can also get for free for 1 year if it's your first time creating one. Create AWS account.

BitBucket account with at least one repository in it.

Code Editor (vs code <

And of course a project ready to be deployed or you can clone this very project (It's nothing fancy , just a simple hello world app in react)

Setting up AWS environment.

For auto-deployment from bit bucket to AWS is done in three stages.

Push the repository code to AWS S3 storage AWS CodeDeploy will pull the code from s3 and deploy it to an EC2 instance EC2 instance is where the code will compile , build and will be served.

Step 1 : Setting Up IAM Account in AWS.

Follow the steps to create a new IAM user which bitbucket will use.

Under "Your Name" menu on top right , select My Security Credentials .

Go to Users tab and click "Add User"

Give an appropriate name.

Make sure to check Programmatic access and NOT AWS Management Console access as these credentials will only be used by bitbucket.

In Add user to group create a new group and from policy list make sure the group has AmazonS3FullAccess and AWSCodeDeployFullAccess checked. Once clicking on the group name , the should look like the following.

Make sure the new created group is checked and proceed and click Create User on the next page.

NOTE : This step is very important as you can do it only once. Click on the newly created user's Secret access key show. And note that key down because we will need it. note down the Access Id Key as well.

Now select Roles from the left navigation bar.

Create role , Select EC2 from the services and EC2 again in Select your use case .

Select AWSCodeDeployRole & AmazonS3FullAccess from policy list.

Give a proper name to the role and create role.

Select the role which was just created , go to Trust Relationship and select 'Edit Trust Relationship' and enter the following in there.

Change the region (us-west-2) according to your region. Just check your url in the address bar. it will be something like.

Step 2 : Setting up S3 Bucket.

Go to Services and select S3 .

Create a New Bucket.

Give a proper Name to it and select a Region. Make sure the region you select is the same as the region you have entered while creating IAM user's role.

Click Create and select the created bucket.

Go to Permission tab and select Bucket Policy and from bottom select Policy Generator.

Enter the details. For Principal enter your IAM User ARN and resource enter your bucket name.

Once done click Add Statement then Generate Policy and copy and paste that into your bucket policy.

Put a /* in Resource after your bucket name and press save.

Step 3 : Setting up EC2.

Time to create an EC2 instance where the real code would be deployed.

From Service menu select EC2 to open EC2 dashboard and select Launch Instance to launch a new EC2 instance.

Select the appropriate Amazon Machine Image (AMI). For this demo I have chosen 'Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-6a003c0f'.

Choose an Instance Type, in free tier you will get 'General purpose | t2.micro' , select and proceed to Configure Instance Details.

In the details , leave everything default or select as per your requirement, but make sure to select IAM role and select the IAM role which we created in IAM settings and proceed.

Add storage, leave everything default and and Add Tag , create a new tag "NAME" and give it a value like 'EC2_INSTANCE_FOR_DEMO_RUN' then proceed to Configure Security Group.

Give a proper Name to the security Group then Add Rule , Select Type : Custom TCP Rule , Port Range : 3000 (for my project) , Source : Anywhere (for the site to be accessible for all user) and finally press 'Review & Launch'

Select a new key pair, give a proper name and download the key and keep it safe. You will need this key file to ssh to your EC2 instance.

Note : You might need to convert the key file to .ppk . Use putty to convert your pem file to ppk.

You need to install code deploy agent in your newly created instance. So, ssh to your instance and run the following commands.

bucket-name is the NOT then name of S3 bucket you created.

For the US East (Ohio) Region, replace bucket-name with aws-codedeploy-us-east-2. For a list of bucket names, see Resource Kit Bucket Names by Region.

To check if the code deploy agent is working or not , run the following command.

If the status is Active then everything is ok else run following command.

Step 4 : Setting up AWS CodeDeploy.

This would be the last setting for AWS.

From the services menu select AWS Code Deploy.

If this is your first application then select Custom Deployment else select Create Application.

Give a proper name and group name to the application and make sure Compute Platform is EC2/On Premisses.

Deployment type : Inplace Deployment.

In Environment Configuration select the tab Amazon EC2 Instances and select KEY , select the TAG KEY and value created while configuring EC2 instance.

If everything worked fine then under Matching instances it will show your EC2 instance and it's current status.

Set Deployment configuration to "CodeDeployDefault.OneAtTime"

In Service role ARN select your "Role ARN". It will be a dropdown so just select the role which you have created in the IAM setting steps.

Click on Create Application.

Setting up Bit Bucket Environment.

Pipeline Environment Variables.

Go to your bit bucket account , open your repository, enable pipelines, then go to settings, PIPELINES and set these environment variables.

Key Value Desc AWS_ACCESS_KEY_ID AKIAJVF47GKFNRKX2X5Q IAM user ACCESS KEY ID AWS_SECRET_ACCESS_KEY ********** IAM user SECRET ACCESS KEY APPLICATION_NAME BitBucket_Deployer_Application Your Code Deploy Application name AWS_DEFAULT_REGION us-east-2 Region of your aws S3 & EC2 DEPLOYMENT_CONFIG CodeDeployDefault.OneAtATime CodeDeploy Deployment configuration value DEPLOYMENT_GROUP_NAME Deployer_APP_GP1 Code Deploy application Group Name S3_BUCKET code-deployer-bucket S3 bucket name.

Setting Scripts - [1] bitbucket-pipelines.yml.

Create or update if already present bitbucket-pipelines.yml and add the following.

Setting Scripts - [2] appspec.yml.

Create a new file appspec.yml in your app's root directory and add the following lines.

Key Value Desc source / Source for Code Deploy to copy files from. Here it's root directory for your S3 bucket destination /home/ubuntu/helloworld/ Destination in your EC2 instance where code deploy will paste and uzip the files BeforeInstall scripts/install_dependencies.sh Script which will be run before your app starts or gets build. Install all the dependencies here like installing node , or npm install etc. Location : yourapp/scripts/install_dependencies ApplicationStop scripts/stop_server.sh CodeDeploy will run this scripts before any other lifecyles hooks. Write your cleanup code like stopping the old server or cleaning older residue files etc. Location : yourapp/scripts/stop_server ApplicationStart scripts/start_server.sh Add your code to run your server or start your application in this script. Location: yourapp/scripts/start_server runas root Run the scripts as given privilege.

Following install dependencies script to install node & npm on EC2 instance.

Following script is to start server or application.

Following is to stop any older instance of node running before start_script runs.

That's pretty much it. Git commit and git push and see the pipeline working.

Quick Setting and Deployment.

Just apply/cross-check the following settings while creating/modifying each one of the entity.

Note : Settings like ARN(s) , ID(s), SID(s), region etc are auto generated. They need not to match the following. Those data are specific to every user.

If there is any problem while applying then please follow the extended version above.

Issues (I've faced) and Fixes.

CodeDeploy Log file location.

Following are the list of issues/errors and their fix which I've faced while getting the code deploy to work. This might come in handy.

1. Delete everything from this path /opt/codedeploy-agent/deployment-root on your EC2 instance and try to deploy again again.

How To Upload And Download Files In Amazon AWS EC2 Instance.

Uploading and downloading files in AWS instance can be done using Filezilla client or Linux scp command. If you are a windows user, you can use WinSCP for transferring files to your EC2 instance.

In this tutorial, I will explain how you can transfer files to AWS instances using the following methods.

Copy Files Using Filezilla Client (GUI Based). Copy files using SCP (Works only on Linux/MAC systems)

Upload Files To EC2 using FileZilla.

You can download FileZilla client from here Download FileZilla.

Step1: Install and open FileZilla.

Step2: Go To Edit–>Settings–>SFTP.

Click add key file and add your .ppk key of your AWS instance and then click ok. You can convert the AWS pem file to ppk using puttygen. Refer this tutorial –> How To Create .ppk File Using PuttyGen.

Step3: In FileZilla homepage enter the host details (public IP, elastic IP or the public DNS) and enter the username in the relevant field. (username varies for different images. For ubuntu Instance, the username will be ubuntu, for Linux machines, the username will be ec2-user) .

Leave the password field empty, since we are using the private key for authentication.

Port number is 22 for SFTP. Once you enter all the necessary details click connect.

Filezilla will be connected to your server instance and you can view your server files and folders.

Step4: File Upload And Download:- Once FileZilla is connected to your server instance, You can upload files to your instance and also you can download files from your server instance.

The local site is your local system files. The remote site is your server instance.

To upload files choose the directory on your server where you want your files to be uploaded.

Now select the file to be uploaded and right-click on it. Click the upload file option. Your file starts uploading to the directory you selected. Similarly, you can download files from your server instance by right-clicking the file.

Few key things to understand,

Since you are logging in as a user, you cannot upload the files to the root directory due to permissions issues. Alternatively, upload the files to the user home directory and copy it to the desired location using any ssh client like putty. You will be uploading and downloading files as a normal user and not a root user. So the files which are owned by root cannot be uploaded and downloaded. You will get permission denied error when you try to download a file owned by root.

So, if you want to download those files, ssh into the machine and change the owner of that file to the normal user using the following command.

In the above command, the user is your instance's default username. For example, for ubuntu instances, it's ubuntu and for RHEL instances its ec2-user . Give the username accordingly. If you want to change the owner of a recursive folder, add a -R switch to the command, as shown below.

Upload File Using SCP Command Line Utility.

If you are using a Linux or Mac system, you can use the scp utility to upload files to ec2.

Let’s have a look at the syntax.

If you want to copy a whole folder, then you should use a recursive flag -r with the command as shown below.

I hope this article helps. Let me know in the comments section if you face any errors.