AWS CLI(Command LINE INTERFACE):

From the beginning, we are using the AWS UI. But this AWS is not automation friendly as UI is not automation friendly. Example: you are requested to create the 10 VPC for a project or 10 EC2 instances for the project, so you will be creating this one by one and your time would be wasted for the creation of resources which would not be efficient.

So, AWS CLI is the solution.

So, AWS comes with APIs through which you can create, delete and manage the AWS resources.

Now, if we want to create the 10 or 100 instances, so we can send the request by creating a shell script to these API and our manual work would be automated.

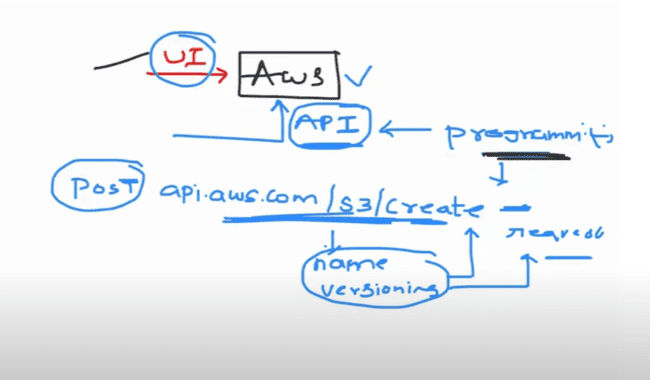
Tools for this:

* AWS CLI
* TERRAFORM
* CLOUD FORMATION
* CDK

The last three are from the class IAC.

AWS CLI is a python utility,

For example: to create a S3 bucket, we can create using the UI or we can call the AWS from the AWS CLI using the APIs through post request of /create/ and pass the request parameter as name and version control.



Now this seems very difficult, as we might need to code in this, to call the API and other stuff, These AWS tools create the abstraction layer over these APIs, which can be called through that abstraction layer.

We need to pass just the name of the instance which we need to create, so we can use that by passing parameters only and this abstraction layer would take care of this thing.

**AWS CLI and AWS CLI Reference DOCS:**

CLI is a layer between the User and API.

A diagram of a program

AI-generated content may be incorrect.

Why there are four terms related with CLI:

If you want the quick information like s3 buckets list, then you can do this using the AWS CLI and you can quickly get that information.

On the other hand, if there is need to large infrastructure and a stack of resources on the AWS, and combination of resources, then CFT and terraform would be fine, and, they act API as code, which helps to review the code as well.

To use the UBUNTU commands, we can use the following on Windows:

A screenshot of a virtual box

AI-generated content may be incorrect.

OR git as well:

A computer screen shot with text

AI-generated content may be incorrect.

Firstly, need to install AWS CLI on the laptop:

A screenshot of a computer

AI-generated content may be incorrect.

Do this:

A screenshot of a computer

AI-generated content may be incorrect.

After installation: run aws on the command line. And also python needs to be installed on the machine.

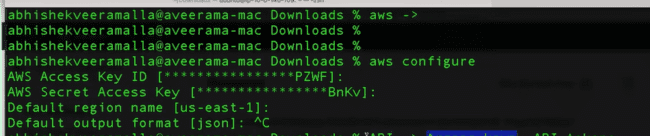
A screenshot of a computer

AI-generated content may be incorrect.

To configure the AWS, we need to use the following command:

aws configure.

But we need the following information and need to configure them in the AWS UI:



User -> Security Credentials:

A screenshot of a computer

AI-generated content may be incorrect.

There are access keys, and you can create only two access keys on the account:

A screenshot of a computer

AI-generated content may be incorrect.

Delete one and create one.

A screenshot of a computer program

AI-generated content may be incorrect.

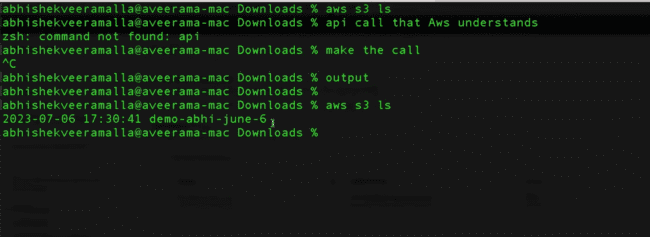
From this access key, we need to configure the AWS:

A screenshot of a computer

AI-generated content may be incorrect.

Use command AWS configure and pass the access keys and then check the access using the following command to check the S3 buckets:

IMPORTANT: when you pass the command on the CLI, AWS translates the command to the API that AWS understands and then make that call.



And same s3 would be there in the UI as well.

A screenshot of a computer

AI-generated content may be incorrect.

TO CREATE THE EC2 INSTANCE:

Following is the command for creating the EC2 instance on the CLI:

A computer screen with green text

AI-generated content may be incorrect.

And it would return the JSON, when create the EC2 instance:

A screenshot of a computer program

AI-generated content may be incorrect.

From the above, we can verify that the private IP is same or not for the above instance created.

A screenshot of a computer

AI-generated content may be incorrect.

AWS CLI reference is the place where you need to check all other commands.

A screenshot of a computer

AI-generated content may be incorrect.

When we miss some parameters while trying to make an EC2 instance, then it would throw the following error which is self-explanatory:

A computer screen with green text

AI-generated content may be incorrect.

**IMPORTANT CASE:** When you are asked to create the VPC, EC2 instance and LOAD BALANCER, then in that case, AWS CLI will not be helpful. These are the complicated tasks, so it won’t be that useful in that part. Then we need to go for the TERRAFORM or CFT.

CFT (CLOUD FORMATION TEMPLATES)

Creating, managing and updating cloud infrastructure on AWS. It is based on the principle of IAC.

Infrastructure as a code means: write code to create an infrastructure. We generally write java code to create the apps, we write angular code to create the web apps. But infrastructure as a code means we write code to create an infrastructure.

Principles of IAC:

IAC tool should act a middleman between user and AWS. User must submit YAML, JSON file or Declarative, Versioned in nature and these templates needs to be converted to APIs that has been understand by the AWS. And the conversion needs to be done using the IAC tool.

**CFT supports YAML or JSON only. CFT only supports AWS but TERRAFORM and CROSSPLANE supports multiple CLOUD PROVIDERS.**

Versioned is same as the GITHUB, where we can check the changes 5 days ago or 6 days ago.

DECLARATIVE:

What you see is what you have.

It means what it should written on the TEMPLATE or on the YAML file, that should be available on the AWS. And this template is easily readable like it would be creating the VPC, RT, LB and EC2.

**IMPORTANT: For listing the S3 buckets in the AWS, if we write templates for that using the CFT, then it would not be good. And in that case, we can use the CLI for quick responses.**

**YAML VS JSON:**

In YAML we can add comments, and YAML is less complex and more readable. And most of the cloud platforms prefer YAML, instead of JSON.

Features of the CFT:

1. Creating infrastructure
2. DRIFT DETECTION: Like you submitted a template for infrastructure creation and then someone unintentionally deletes something or disable the versioning. So you can use the detect drift option in the CFT and you’ll be notified for the change and you can take the necessary action.

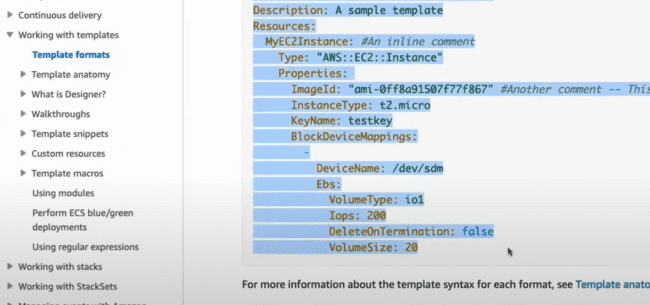
**Now you have written the YAML file on personal laptop and want to submit the same to the CFT, so that thing you can do using the STACKS which imports the template to the CFT.**

CFT has a proper structure for writing the YAML.

1. Version
2. Description
3. Metadata: who is the owner of the CFT.
4. Parameters: variables which need to be passed.
5. Rules: validating the parameters. Like, when you create a S3 bucket, then rule can be, the name of the bucket should be like that it should provide information that in which environment it is getting created in production or staging.
6. Mappings: assigning the parameters to the variables.
7. Conditions
8. RESOURCES: this is mandatory. This is like which thing you want in the template. Like do you want to create the EC2 instance, S3 instance or anything else.

READING MATERIAL: AWS CFT on Google.

Template format and Template anatomy are main things, anatomy is the meaning of each thing used in the template.



First line mentioned the name of the resource.

In the resources, type is the main parameter through which CFT understands what it needs to create.

We can pass parameters at the run time to make the more than 1 instances in AWS. Parameters like image id, instance type, key name. This is the main use of the **PARAMETERS.**

**RULES:** For the above template, if that is shared with multiple people across the team, then someone requests for the instance type as t2.xlarge, then cost would increase. So, to avoid this, we can use the rules to ensure that, instance type can only be t2.small or t2.micro, for these **INSTANCE TYPE PARAMETERS.**

**MAPPINGS:** If you want to pass the t2.micro as a variable and want to read this in the **INSTANCE TYPE PARAMETER,** then you need to map this variable at the run time.

**CONDITIONS:** If you want the CRT template to run in the DEV environment only, and not in production, but someone tries to run that in the production, then you can take any **PARAMETER** from the above template and make sure that should run only on the DEV environment and not in the PRODUCTION.

**TRANSFORM…. LATER**

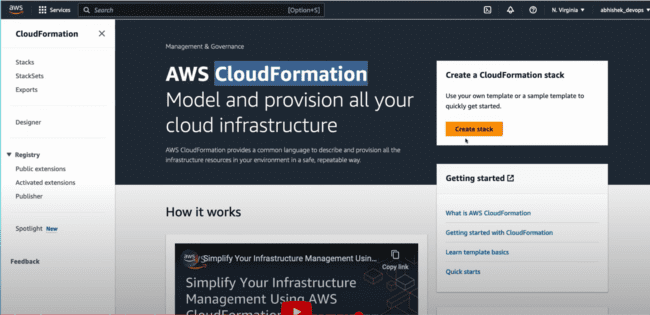
**RESOURCES:** this field is mandatory.

**Under Resources, you can pass group of resources, as everything which pass inside resources is a OBJECT type. And you can pass as many resources as you want.**

**OUTPUT:** We can provide the output, like which type of output we want in our case. Which can be instance id, private instance id, image id, or full JSON, we can provide that in the **OUTPUT** section.

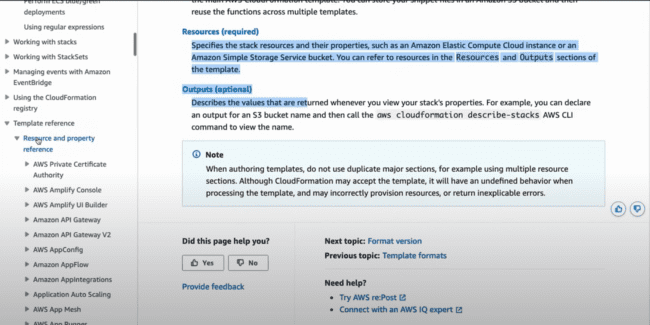
Writing templates:

Search for cloud formation -> create stack.



Template needs to submit to the stack; stack convert the template to the API using cloud formation services.

In the documentation, under Template reference-> Resource and property reference-> They have all the resources needed to create the resources for the AWS and samples as well.



For the Use a sample template, we can select the template from the predefined templates.

We’ll be using the following format for now:

A screenshot of a computer

AI-generated content may be incorrect.

This is the tool which would be opened when we click on the “Create templated in designer”, which is a DRAG AND DROP tool.

A screenshot of a computer

AI-generated content may be incorrect.

Now if you want to use the operations on the bucket drop the bucket from left and drop on the right side and code would be generated below:

A screenshot of a computer

AI-generated content may be incorrect.

You can switch in the YAML format also as:

A screenshot of a computer

AI-generated content may be incorrect.

We need to provide the type necessary to the element, name can be any:

A screenshot of a computer

AI-generated content may be incorrect.

Now from this one, if we want the name of the bucket, we can use them in our template.

A screenshot of a computer

AI-generated content may be incorrect.

And use the same in the designer:

A screenshot of a computer

AI-generated content may be incorrect.

After the Cloud formation -> Create stack.

A screenshot of a computer

AI-generated content may be incorrect.

Either we need to provide the Amazon S3 URL or upload a file.

We’ll be uploading the file from the desktop saved as .yaml file.

A screenshot of a computer

AI-generated content may be incorrect.

Click on the next and provide the name:

A screenshot of a computer

AI-generated content may be incorrect.

After confirmation, resource creation is in progress.

A screenshot of a computer

AI-generated content may be incorrect.

Bucket under S3 would be created:

A screenshot of a computer

AI-generated content may be incorrect.

The bucket starting with cf-template is created by AWS itself and all the cloud formation template configuration would be inside this one:

A screenshot of a computer

AI-generated content may be incorrect.

Delete the created bucket. And check in the stack for the drift who has deleted that bucket.

A screenshot of a computer

AI-generated content may be incorrect.

On clicking the Detect stack drift, we are able to check that something has changed as status is drifted, but as the bucket gets deleted, so nothing was mentioned.

A screenshot of a computer

AI-generated content may be incorrect.

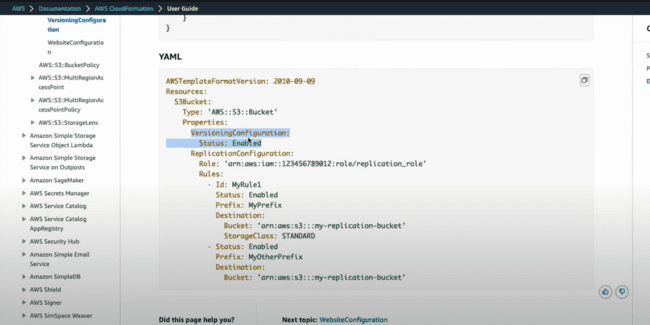
Delete this stack and check other case who has done any changes in the cloud formation.

A screenshot of a computer

AI-generated content may be incorrect.

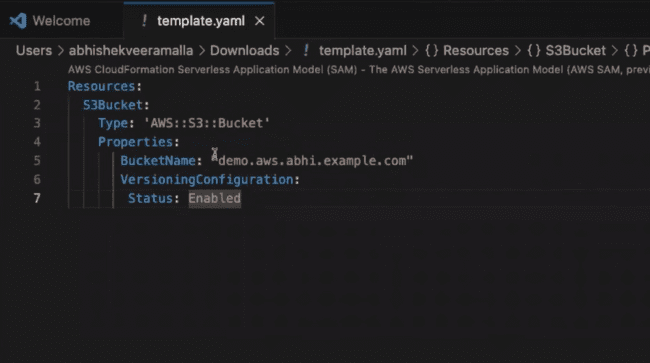
**CASE: We’ll be creating a template which has bucket versioning enabled, and through manual entry, we’ll disabling the S3 versioning and check how CFM drift behaves in this case.**

Versioning configuration, we need to check:



Same property we need to use in the YAML file.

Through the following configuration, create the S3 bucket using the same steps:



**Bucket is created:**

A screenshot of a computer

AI-generated content may be incorrect.

Change the versioning in S3 bucket:

A screenshot of a computer

AI-generated content may be incorrect.

To disabled:

A screenshot of a computer

AI-generated content may be incorrect.

After changes, Go to the Stack and detect drift:

A screenshot of a computer

AI-generated content may be incorrect.

Detect drift.

A screenshot of a computer

AI-generated content may be incorrect.

You can see the same in the Resource drift status:

A screenshot of a computer

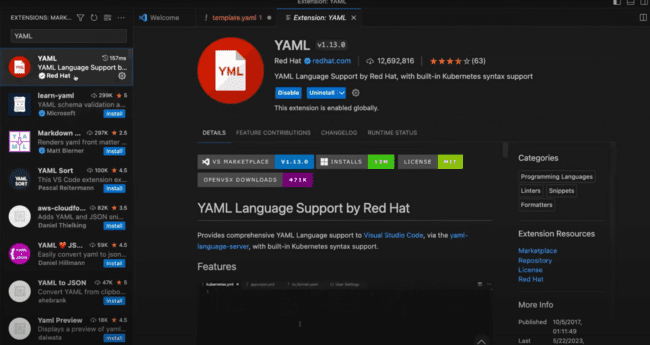
AI-generated content may be incorrect.

Click View drift details: It shows the status is Expected and the Actual.

A screenshot of a computer

AI-generated content may be incorrect.

Plugins for writing the templates easily:



A screenshot of a computer

AI-generated content may be incorrect.

HELPTEXT with the AWS toolkit:

A screenshot of a computer program

AI-generated content may be incorrect.

*CFM vs TERRAFORM: Terraform is good for hybrid or multi-cloud environments, that uses both AWS and AZURE, as it can handle those things. Terraform is the tool for the future, there are many advantages of Terraform over CFM, and there is one technology, CROSSLANE may take over terraform in future.*

**AWS CI/CD, AWS CODE COMMIT:**

There are following tools available in the AWS for complete CI/CD.

1. AWS Code Commit
2. AWS Code Pipeline
3. AWS Code Build
4. AWS Code deploy

A screenshot of a computer

AI-generated content may be incorrect.

Now we need to do the code and push the code to the GitHub and then there is hook used in the GitHub, which triggers the job in the Jenkins, which would trigger the CI pipeline, and then code build process takes place, like building, code review using SonarQube and then build is deployed on the Kubernetes.

1. GitHub for hosting the code.
2. Jenkins as orchestrator and for the pipeline
3. Build process using Docker and SonarQube for integration.
4. Argo CD or shell script for deploying the application.

These all things offer by the AWS tools mentioned above.

**AWS CODE COMMIT:**

Advantages:

Mostly organizations use enterprise GitHub or make their private repository on the GitHub. Or, they install them on their servers and might be they also need to scale their servers or need to install more number of servers as well.

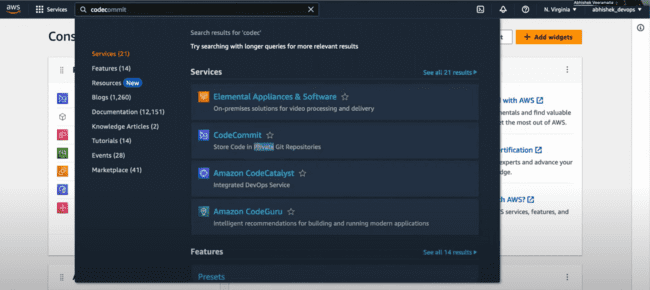
So, the solution provided by AWS is AWS Code Commit which is a managed service.



Now, if there are number of applications increase, then you need to git repositories, then there would be an overhead.

So, Aws said to create as many as repositories as possible, they will manage the creation of EC2 instances and scalability, and you just need to pay them for their services.

Search for the Code commit in the AWS:



The repositories created inside the AWS are private by default.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

There is a warning sign that not to use the root user:

A screenshot of a computer

AI-generated content may be incorrect.

Clone URL, PR, Commits, Branches all are here:

A screenshot of a computer

AI-generated content may be incorrect.

We can add the file from the UI as well as from the command prompt as well.

A screenshot of a computer

AI-generated content may be incorrect.

Templete.yaml file has been uploaded from here:

A screenshot of a computer

AI-generated content may be incorrect.

After providing the author’s name and email address, we can do the commit:

A screenshot of a computer

AI-generated content may be incorrect.

**IMPORTANT: One thing is that we can only commit the single file from here only, if we want to commit multiple files, we need to use the command prompt.**

Code commit needs to be done from the IAM user only, and we need to attach permissions as ***AWSCodeCommitPowerUser*** only.

A screenshot of a computer

AI-generated content may be incorrect.

Now after login from this user, we need to use the same repository and made the code changes as well.

Other steps are same as the command prompt.

* git clone <URL>
* git checkout <repo-name>
* git commit
* git push

**Need to check the usage of git config….?**

**DISADTANGES OF CODE COMMIT:**

GitHub has copilot integrated with that; GitHub is coming with new integrations with visual studio code.

You can edit the code online using the visual studio code using the GitHub. So, GitHub and Gitlab are rich in nature and has a lot of new features.

While AWS code commit is very restricted. And all the integrations are with AWS only and very less integrations with services outside AWS.

**AWS CODE PIPELINE:**

Code is committed to GitHub, and when code is committed, then a web hook is invoked.

Then, continuous integration takes place which involves steps:

* Check out
* Build and unit testing
* Code scan using SonarQube
* Docker image creation
* Docker image scan
* Docker image push

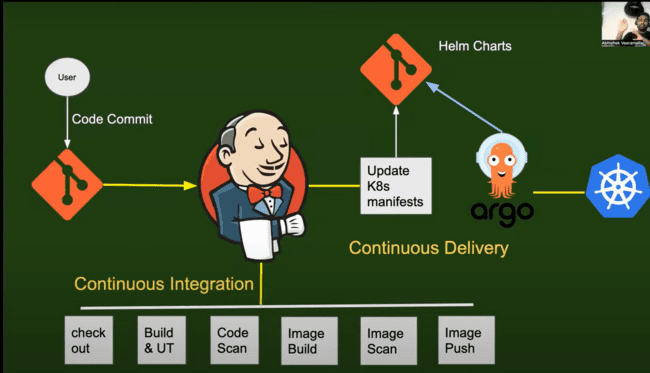
**After this CD (Continuous delivery) stage invokes. You can use the ANSIBLE or shell script, which is now a days an outdated way. In this, we write a Kubernetes YAML file and push the same to the Kubernetes cluster.**

**Or Helm chart can be created using the shell scrips or python scripts and use the cube CTL or HELM command to push the same to the Kubernetes cluster.**

**And if there are multiple Kubernetes cluster, then multiple Kubernetes cluster are configured as host and that can be push using the ANSIBLE Play hook.**

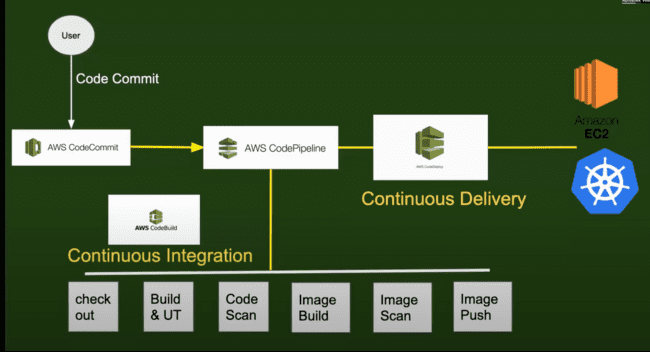
**But the recommended approach is GitOps which not only pushes the artifacts or Kubernetes related resources to the Kubernetes cluster, but you can also constantly manage them. Because they can make sure that whenever someone makes any changes or modify any resources on to the Kubernetes cluster, it can revoke the change, or it will put back the change to the single source of truth which is a git platform.**

**Or ARGOCD which is GitHub solution can be used in this which is now a preferred solution and mostly used now a days.**

****

How CI/CD works in the AWS….?

AWS Code Pipeline works same as that of Jenkins. AWS Code Build takes care of all the Continuous Integration and AWS Code Deploy takes care of the Continuous Delivery and then delivers to the Kubernetes cluster or EC2 instances.



**Why AWS pipeline is preferred…...?**

You have installed Jenkins and for Jenkins we need to follow master-slave architecture. The slaves can be DOCKER agents or the virtual machines.

For 50 Jenkins pipelines, you started with 2 worker nodes, then you started increasing the worker nodes to 4, 8, 16, 20 and soon. And then you need to take care of these all 20 machines like managing, scaling, security patches, nodes are healthy or not, add resources of nodes.

So, there we need to have one dedicated DEVOPS engineer which needs to take care of all this managing part.

So, that’s why they came up with the solution of these AWS CI/CD pipeline.

But sill why JENKINS is preferable in this part. The big companies can put the entire workload on two worker nodes or on two EC2 instances and if they are managing this efficiently using the docker slaves or agents, in that case, AWS CI/CD is not required.

Also creating the JENKINS on the EC2 instances, auto-scaling groups, AWS cloud watch, AWS alarm, and the AWS would take care of this one as well and then you can easily scale these things up.

Also, AWS pipelines are only restricted to AWS, so if the company wants to move to other cloud or to the hybrid cloud, in that case also, the whole pipeline needs to be updated or discarded, as AWS pipeline works with AWS only.

Now in case of Jenkins, if the Jenkins pipeline is created on the EC2 instance, and by tomorrow the company wants to move to other cloud, in that case, we can take the backup of pipeline and creates the Jenkins instance somewhere else.

**CREATING THE CI PIPELINE USING THE AWS:**

AWS Code Build: Search for AWS Code Build and use the IAM account, preferably.

The option of “Restrict number of concurrent builds this project can start” says if you want to restrict the users sending out concurrent pipelines. Example: if 100 users are submitting 100 builds concurrently, then do you want to restrict them or not.

A screenshot of a computer

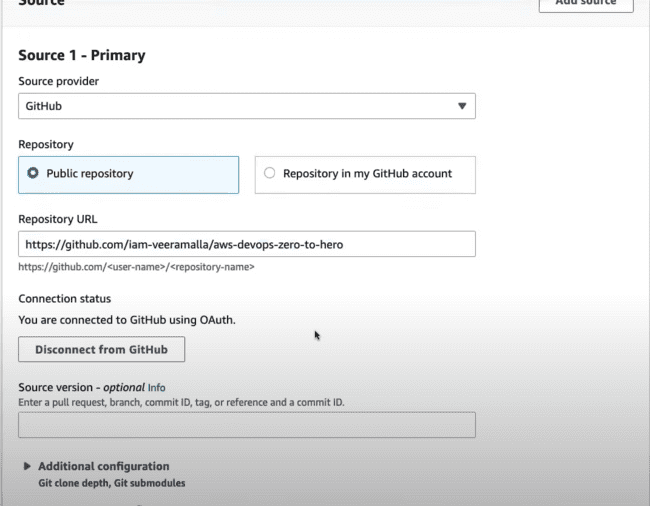
AI-generated content may be incorrect.

Primary source: these can be used, but if you want any other to use, like: Gitlab, then you need to mirror the same in any of them following and then use them. Like mirroring the Gitlab in the Amazon S3 and then use.

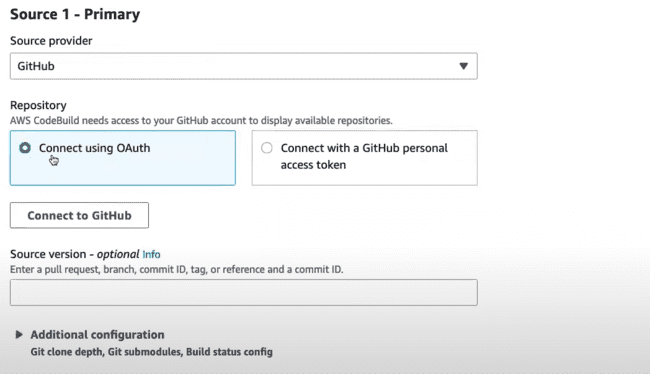
A screenshot of a computer

AI-generated content may be incorrect.

Any public repository in the format given under the “Repository URL”:



Before connecting to GitHub, the primary source would look like this:



After connecting using the “Connect using OAuth”, it would look like this:

A screenshot of a computer

AI-generated content may be incorrect.

**AWS Code Pipeline invokes the AWS Code Build and code build provides the environment for the code build which would be VM’s or Docker Image. And AWS code build provides you readymade images and you can run your build on those images.**

Use the following environmental image:

A screenshot of a computer

AI-generated content may be incorrect.

IMPORTANT related to ROLES:

*To perform the actions on the AWS Code Pipeline, we need to have service role or an IAM role.*

*Rule is required here as, we are using some service which is performing some action not a user, that’s why we need* ***IAM role*** *here.*

Service Role:

A screenshot of a computer

AI-generated content may be incorrect.

Buildspec is the main part:

All the tasks like build, creating image or push needs to be done using this buildspec file.

We can use the pre-defined YAML file or use other option to write using the Editor and AWS will help in this:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

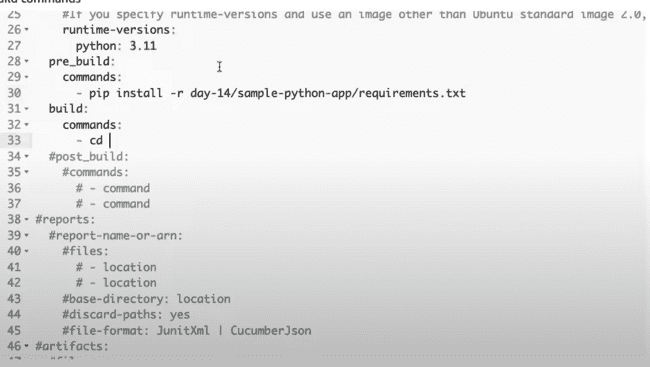
AI-generated content may be incorrect.

Whichever phase we need to write, we can uncomment that and start writing.

Pre-build is which steps you need to perform before building the application. Like you need to install something or need to install something on the image while building the application.

And need to provide the absolute path where that file is place, which you are referring to.

Example: The pre-build command.



And the requirements.txt contains only flask keyword:

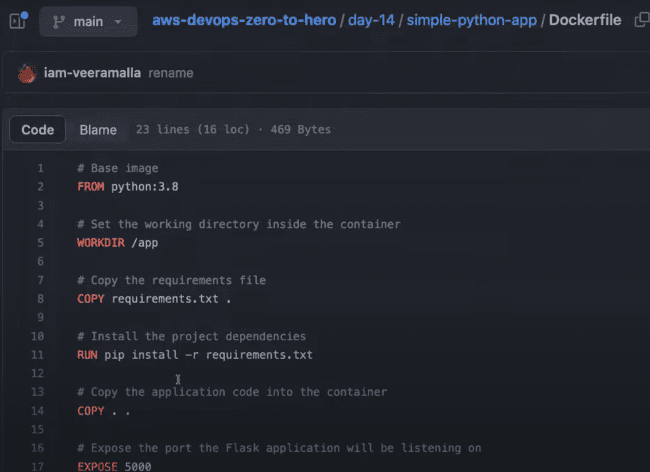
A screenshot of a computer program

AI-generated content may be incorrect.

As the GitHub repository has been integrated, the checkout would be taken care by the AWS itself, so we can skip this step.

For building switch the directory and then build the docket image:

The docker file in the project:



We need to store the username and the password somewhere in the AWS system administrator to pick from there, as that is very sensitive information.

We need to use the system manager to use the user name and password in the file which is now left with “<>” only.

A screenshot of a computer

AI-generated content may be incorrect.

After post build, print the message:

A screenshot of a computer program

AI-generated content may be incorrect.

After that create build. But service role was missing.

A screenshot of a computer

AI-generated content may be incorrect.

Creating a new service role.

A screenshot of a computer

AI-generated content may be incorrect.

Code build we need to select here:

A screenshot of a computer

AI-generated content may be incorrect.

Click Next and grant permissions later and provide the role name and create the role.

A screenshot of a computer

AI-generated content may be incorrect.

Now use the same role there after refresh:

A screenshot of a computer

AI-generated content may be incorrect.

Service would be created:

A screenshot of a computer

AI-generated content may be incorrect.

HOW TO STORE THE SENSITIVE INFORMATION:

AWS system manager:

A screenshot of a computer

AI-generated content may be incorrect.

Create parameter:

/myapp/docker-credentials/username: format would be easy to remember for specific application.

A screenshot of a computer

AI-generated content may be incorrect.

Use the KMS as encryption and store the value.

A screenshot of a computer

AI-generated content may be incorrect.

Under AWS Code Build, edit and store the username and password.

Code build -> Build details -> Edit buildspec file.

Uncomment the env: parameter store and pass the values.

A screen shot of a computer

AI-generated content may be incorrect.

And change in the build path also:

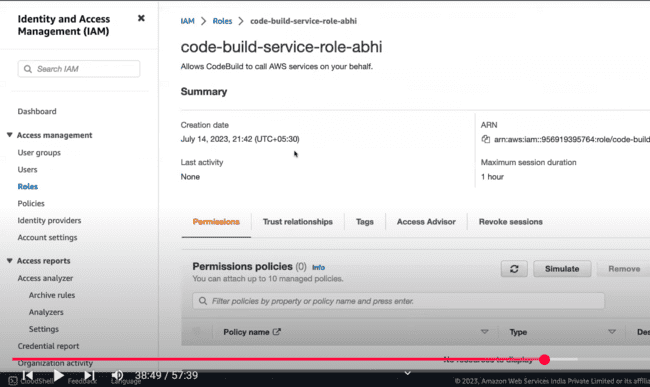
Syntax:

*-docker build -t “<DOCKER REGISTRY URL>****/****<DOCKER USERNAME>****/****<NAME OF APPLICATION>****:latest****”*

*A screenshot of a computer

AI-generated content may be incorrect.*

Click on the start build. And before that we need to provide the access to the role that has been created.



Grant full access to that role:

A screenshot of a computer

AI-generated content may be incorrect.

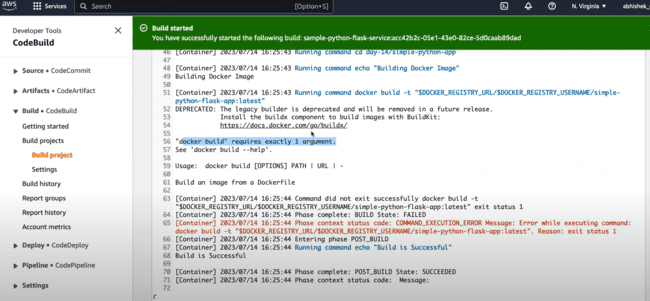
After this build the same, and we can see that there is no file:

A screenshot of a computer

AI-generated content may be incorrect.

Name was wrong, which we can change and start the build. As the Code Pipeline is not integrated, so we are doing it manually, otherwise, that would be automatically configured. Need to replace the word “sample” with the “simple”.

Again, there is one issue with the docker command which is as follows when we run the build:



Go to the project and edit the build details:

A screenshot of a computer

AI-generated content may be incorrect.

Go to the build details and edit the sample python service:

A screenshot of a computer

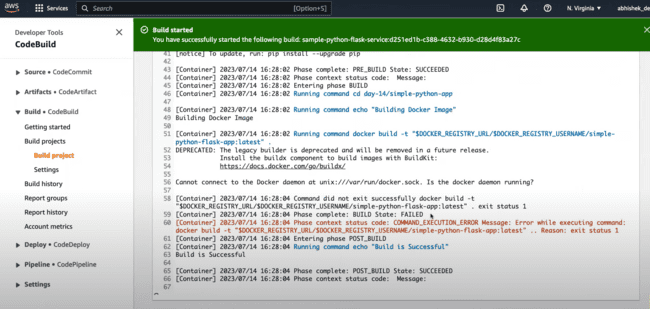
AI-generated content may be incorrect.

The issue is with adding the “.” at the end of the docker command.

**A screenshot of a computer

AI-generated content may be incorrect.**

Again, click on the Start build. And, after starting the build there is one more error which is mentioned as below:



Now this issue can be easily fix giving the additional permissions to the code base. Go to the Project-> Edit -> Environment.

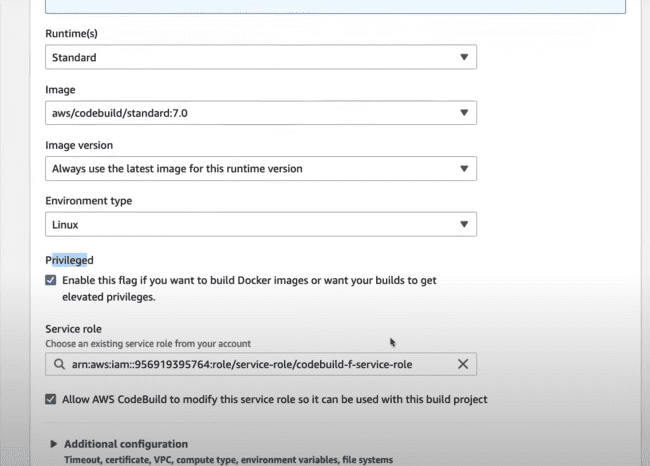
A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

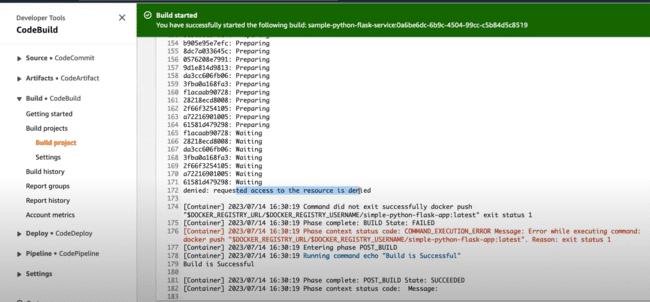
AI-generated content may be incorrect.

*CAUSE OF THE ISSUE: AWS does not allow to create the docker images on the build systems which it provides.*

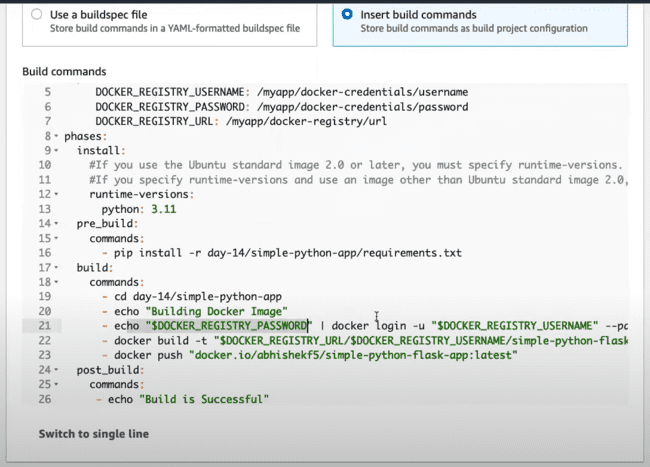
Click on the Override image and you can select the “Privileged” checkbox. And then click on the Update Environment.



Again, after starting the build, the following error comes on the page as:



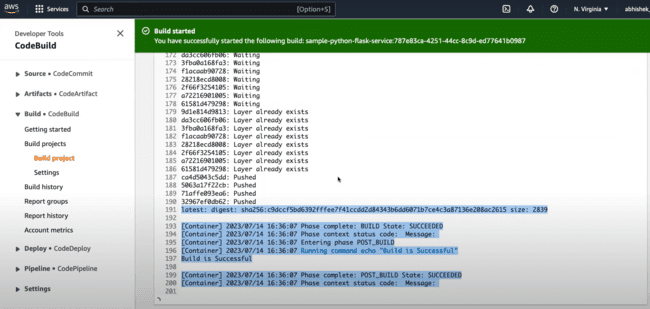
Then again edit the buildspec and check that login command was missing from the docker file.



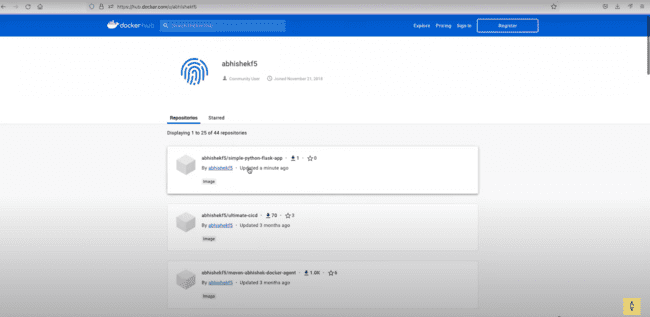
A screenshot of a computer

AI-generated content may be incorrect.

After building again, there is success:

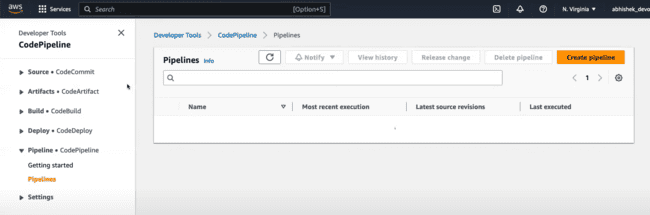


If you login to the docker hub, then you’ll be noticing that, there would be docker image created for the same.



Now, instead of starting the build manually, we need to integrate the same with the webhook kind of thing in the AWS, so that pipeline would be invoked automatically and docker image would be created.

Search for CodePipeline -> Create pipeline.



After clicking the create pipeline. Click Next.

A screenshot of a computer

AI-generated content may be incorrect.

Select source provider:

A screenshot of a computer

AI-generated content may be incorrect.

Select GitHub V2 and connect to GitHub.

Select repository name and the branch name.

A screenshot of a computer

AI-generated content may be incorrect.

Build provider and project name would be:

**A screenshot of a computer

AI-generated content may be incorrect.**

CD is not there right now, so we can skip this thing:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Click on the Create pipeline.

A screenshot of a computer

AI-generated content may be incorrect.

Pipeline created:

A screenshot of a computer

AI-generated content may be incorrect.

On updating anything in that repository, that build would be triggered.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Once it completed, docker image would be created.

A screenshot of a computer

AI-generated content may be incorrect.