

2022-04-04 Initial version A Gonzalez

Document History

Cloud management with AWS CloudFormation update and Application deployment



This document is copyright protected. No part of this document may be translated, reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Enzosystems.

Although every precaution has been taken in the preparation of this document, Enzosystems assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the information contained in this document.

Enzo, the Enzo logo, are trademarks of Enzosystems BV. All other brand and product names mentioned in this document are trademarks or registered trademarks of their respective companies, and hereby acknowledged.

Enzosystems B.V. | DeBinderij 69 | 1321 EC Almere | The Netherlands

[www.enzosystems.com](http://www.enzosystems.com)| [info@enzosystems.com](mailto:info@enzosystems.com) | T. +31 (0) 36 5461040

Table of Contents

[1 Introduction 3](#_Toc27484195)

1. Introduction

This document describes how to use the CloudFormation model and template along with the AMI and the needed tools for the management of the Cloud stacks and the application deployment.

VPC Model and templates:

* We have 1 template model file for building and starting a vpc. From this model file we can create cloudformation template files of a vpc stack.
* The vpc model file take only one parameter the vpc name.
* Once we have a template file (defining a build vpc) this template file can be uploaded in AWS CloudFormation which verify the file and build the vpc. If a template file and its stack with the same name already exist, then the template file and the stack resources are updated, otherwise they are created.

Example:

We want to create a new vpc called ‘developmentVpc’.

* We have 1 template model file for starting a set of ec2 running an application , from the model file we can create a cloudformation template file of an application stack in a vpc.

The generated template file will set up the stack for an application

* The ec2 model file takes few parameter the vpc name, the application name , the application port , the subnets , the key name for the ec2 (optional).

The models files are available in enzo-n1 and in an S3 bucket.

The template files are available in an S3 bucket and cloudforrmation , a backup is in enzo-n1.

The model files must stay the same, they ar e the base structure for the aws stacks. You should always work with a copy and never modify these files directly unless you wan to change the stack base structure.

There actually 2 available bash scripts to help filling the template file and deploying an application for the application stack.

The vpc script take a vpc name as argument, it generate a template file from the model

The steps triggered by the ec2 script are:

1-create an AMI from the github repository

2-create or update the cloudformation stack for this application ec2s

* 1. Tools needed to use the scripts

AWS CLI

Packer (for ec2 deployment script)

* + 1. Preparation
* Install the aws cli (see the doc in enzo-n1\\D2002 Enzo webservices\AWS CLI Installation.docx )
* Upload locally the content of one the folder from enzo-n1\\D2002 Enzo webservices\EnzoWebservices - Cloudformation\Cloudformation\_newFiles\CloudFormationScripts either the VPC stack folder or the APP stack folder according to your needs
* You can create or update a CloudFormation stack :
* 1 - manually, or, 2 - using the provided bash scripts.
  + 1. 1 - To update manually:
* you need to retrieve the CloudFormation template file describing the stack you want to update. You can get it from:

->The ***CloudFormation console*** (**WARNING** : copy or download the template locally on your machine, **don’t modify it directly in CloudFormation** to prevent unwanted modifications of the AWS environments.)

->The S3 bucket named ***cloudformationtemplating***

->From ***enzo-n1\\D2002 Enzo webservices\EnzoWebservices - Cloudformation\Cloudformation\_newFiles\TemplatesBackup*** (the deploy\_stack scripts try to store a backup file of the template in that directory when it runs).

Once your template is filled, you can submit it to CloudFormation using ***uploadAndUpdateStackFromTemplate.sh*** available in enzo-n1\\D2002 Enzo webservices\EnzoWebservices - Cloudformation\Cloudformation\_newFiles\CloudFormationScripts

If you want to completely update the stack it might be then better to start from scratch with the scripts and provide the exact same stack name, use the exact name of the stack and remove the prefix :

-If the stack name is vpc-dev use the name dev

-If the stack name is ec2-messaging-dev use the name messaging

Note: Deploying an application manually must be done once the ec2 stack is deployed , as it cannot be automated this way.

* + 1. 2 - To update automatically :

Same Naming Syntax :

the stack name, is build this way :

- I want to create/update a vpc named Dev :

I use the deploy\_vpc script, providing the name Dev.

This will build or update a stack named vpc-Dev, and build a vpc with the name Dev

* I want to create/update 2 ec2s running the Booking app in the Dev vpc .

I use the deploy\_ec2 script, providing the environment name Dev and the app name Booking. This will build or update a stack named ec2-Booking-Dev, and build a 2 ec2 with the name Booking in the Dev vpc

* + 1. To deploy an application with the ec2 stack :

Prepare:

* an AWS access
* and its Aws secret key,
* a git repository url
* git credentials,(user/pwd)
* the application startup command.

You can use these values to fill the packerEc2AMIBuilderParameters.json file. If you don’t prefilled this file , use the interactive script to provide them during run time.

If you use the deploy\_vpcStack.sh , the parameter is :

* the vpc/stack name, if not provided you are prompted for it

Usage:

.\deploy\_vpcStack.sh Dev

If you use the deploy\_ec2Stack.sh , the parameters are :

* the vpc name,
* the subnets to use ( 1, 2 or 3 ) if not provided you are prompted for it
* the Application/ec2/stack name
* the port for the application to run in and to be open in the Security group, and use for load balancer and target groups (note the port 22 is open by default for ssh)
* the load balancer listener path
* the ec2 keyPair name , if not provided the application name is used

Usage:

.\deploy\_vpcStack.sh Dev 2 BookingSystem 3000 api/booking Booking

You will be prompted if you want to deploy an application with the ec2s. In that case, you need in addition to fill the packerEc2AMIBuilderParameters.json file with:

* an AWS access
* and its Aws secret key,
* a git repository url
* git credentials,(user/pwd)
* the application startup command
* the ec2 type

The app name and the env are filled automatically by the script

Note, if you didn’t filled this parameter file , say no when you are asked about it by the script, this way the script will ask you to provide the values and will use and fill a similar file packerEc2AMIBuilderParameters\_interactive.json

**ATTENTION**,

**Do not fill the packerEc2AMIBuilderParameters\_interactive yourself ,** this one is meant to be filled automatically from the scripts.

**Use packerEc2AMIBuilderParameters.json for manual filling** of the app deployment parameters.

The deploy\_ec2Stack\_interactive.sh is the same than the deploy\_ec2Stack.sh excepted that it don’t expect argument and only ask you to provide the parameters one by one at run time in an interactive way .

Ec2 stack deploy steps:

* the script will ask you for the ec2 stack parameters
* the script will ask you if you want to deploy an application from git
* It asks if all the parameters for deploying the app (packerEc2AMIBuilderParameters file) are filled.
* If no it will prompt for these values and fill the file
* If all ready start the deployment with the AMI creation
* set an ec2, install git , node and pm2
* clone the git repository
* install the dependencies
* set up pm2
* generate an AMI from this ec2
* update the cloudformation model with the provided values
* copy a backup on enzo n1
* upload to s3
* submit to cloudformation
* create or update the selected stack
* create or update the associated AWS resources

1. Cloudformation flow

CloudFormation being used to manage infrastructure, we can easily add , remove , update aws resources by updating the file. In that sense it can be used to update an ec2 running an applications under the condition that we use custom AWS AMI images provisioned with our application code for our deployments .

An automated deployment of an application using CloudFormation and AWS AMI is not the most straightforward. The steps/flow is as follow:

1. Create a new AMI provisioned with the updated code and Pm2
2. retrieve the generated AMI-id.
3. Push the AMI to AWS
4. Update the cloudformation file (in the correct way and section) with that newly generated AMI-id
5. Submit the updated cloudformation file to AWS

Extract of an ec2 description in CloudFormation

"Ec2InstanceName":{

"Type" : "AWS::EC2::Instance",

"Properties" : {

"AvailabilityZone" : "eu-west-1a",

"ImageId" : "ami-0a6610cc81aa75bbf",

"InstanceInitiatedShutdownBehavior" : "stop",

"InstanceType" : "t2.micro",

"KeyName" : "WebService1",

"NetworkInterfaces" : [ {

"AssociatePublicIpAddress" :false,

"DeleteOnTermination" : true,

"Description" : "Network interfaces for Webservice instance 1 (eth0).",

"DeviceIndex" : 0,

"GroupSet" : [ {"Ref": "PublicSubnets"} ],

"SubnetId" : {"Ref": "Subnetv1"}

}],

"UserData": "IyEvYmluL2Jhc2gKc3VkbyBub2RlIC4vaG9tZS91YnVudHUvRGV2ZWxvcG1lbnQvRW1haWxTZXJ2aWNlL2FwcC5qcyAm"

}

},

Highlighted are the part that are likely to be updated during a new application release and deployment using AWS AMI images .

Using a shell script we will update the file and use aws cli command to update the stack, either using change-set functionality if the resource support it, otherwise directly modifying the template file.

1. Packer

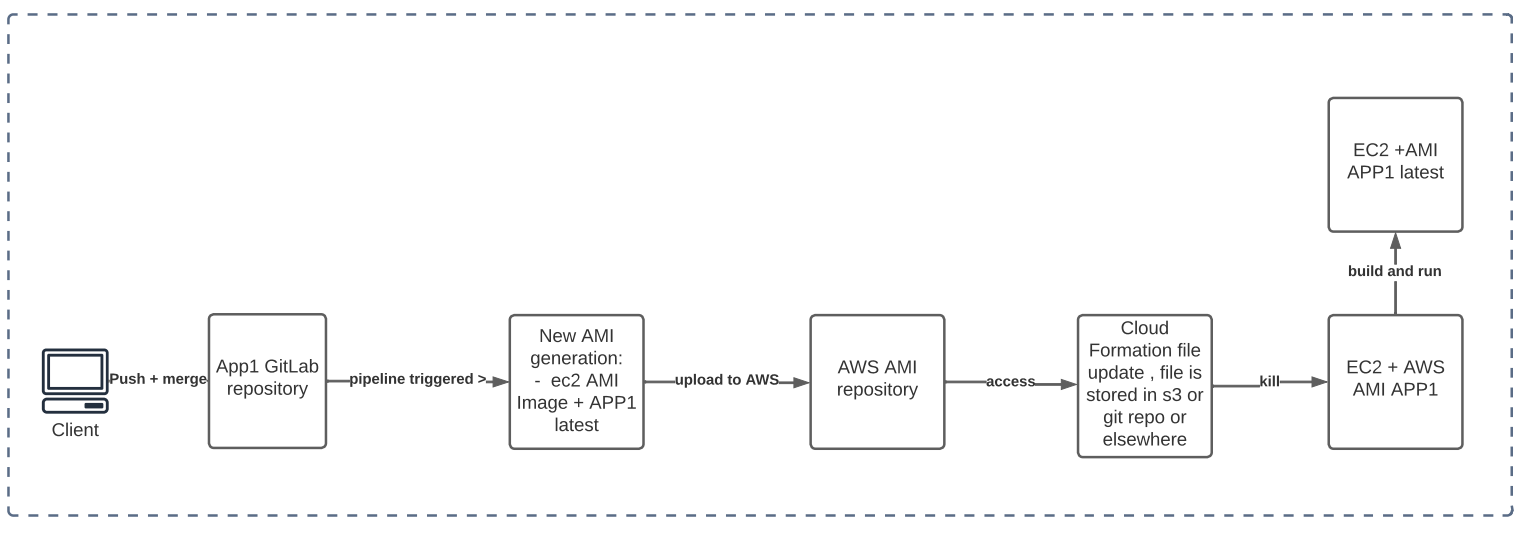
To generate the AMI we can use the AWS console , the AWS CLI or an application named Packer. Packer is an application used to generate various Image types such as Docker Image , AWS AMI image ….

After a code repository update, we trigger a script that will use packer and provide it with various parameters as inputs . This step is to use the code from the updated repository, generate a new AMI from it, and push it to AWS.

This new AMI is uploaded to AWS and available as part of the list of available AMI. And we retrieve the ami id .

We then use this ami id to update the cloudformation file and submit it to AWS.

If all succeed and get validated by AWS, the stack is updated and the ec2 is replaced by a new one using the specified AMI id.



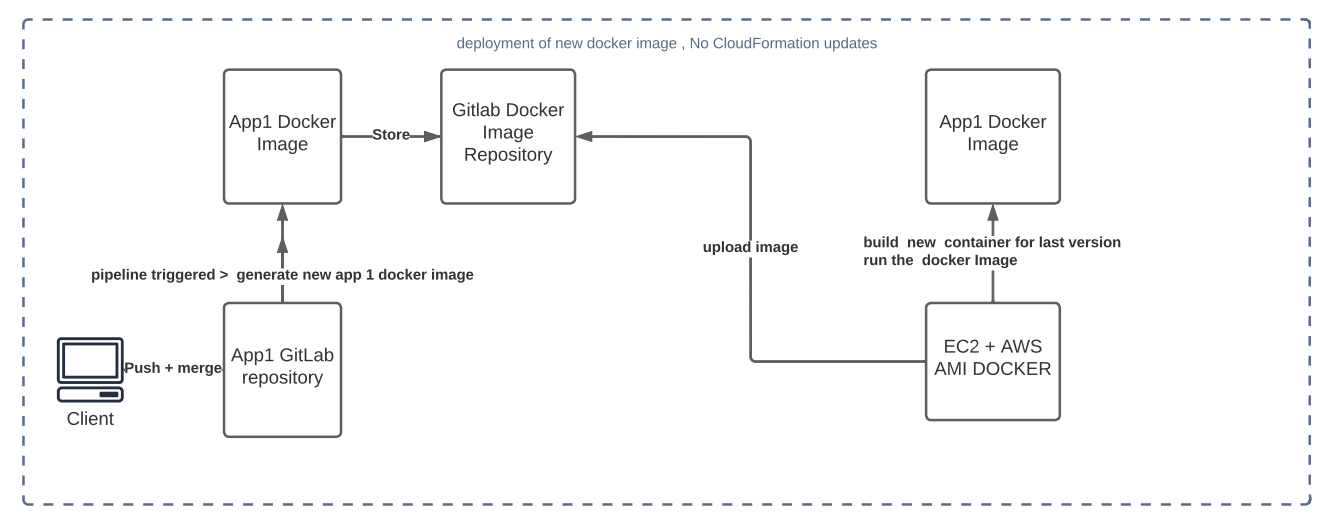
1. Alternative using Docker Image for the application and CloudFormation for the Aws architecture:

Another method is to keep the cloudformation stack managing the infrastructure but not the applications hosted and run on this infrastructure. We won’t be using custom AWS AMI provisioned with our application code but docker images and the docker service available on the ec2 to run the applications. Also instead of the AMI to store our application images we will use a docker image repository.

In that case we use CloudFormation only to manage the infrastructure and the servers running the docker images.

The application layer is provided by the docker service running on the ec2 or on an AWS lambda.

This way a new Docker Image generation is triggered after a code repository update and we restart the docker service running this specific application to load and run the last image of the application.



Note:

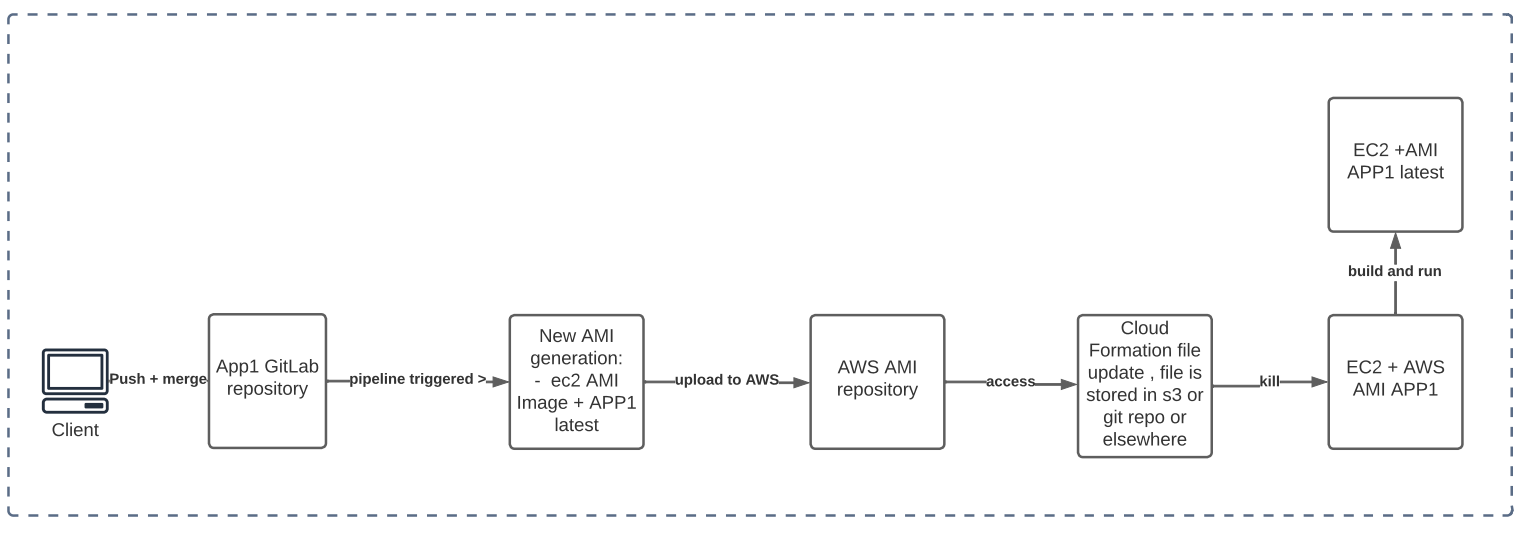
The docker service can also be started and the image to run to be specified in the cloudformation userdata property of the ec2 descriptions. But as the ec2 don’t need to be recreated in that case is better not to include this data in the CloudFormation template and keep it to describe the architecture only. This is a good way to decouple the application layer from the architectural layer.

Updates and maintenance is easier, but that lead to different technologies usage for achieving the same purposes.

This also allow to make the child cloudformation templating more generics and reusable.

1. Method Compare

Application Update flow using Cloud formation



Application Update flow using Docker

