 **While1 Project**

**Architecture Low Level Design**

**Version:** <0.1 27/07/2022>

**Customer**: While1

**Objective:** the goal of this document is to describe the infrastructural architecture realized for the customer by Criticalcase on AWS Cloud.

**Limits**: the application environment is out of the scope of this document.

Summary

[1 Project scope 3](#_Toc109808335)

[2 Cloud and Infrastructure Services 4](#_Toc109808336)

[2.1 Infrastructural Scope 4](#_Toc109808337)

[2.2 Architecture 5](#_Toc109808338)

[2.3 Codepipeline 5](#_Toc109808339)

[2.3.1 Pipeline deploy code 5](#_Toc109808340)

[2.3.2 Pipeline create ami 6](#_Toc109808341)

# 

# Project scope

The main goal of While1 is to upload the code to the EC2 servers via an automated pipeline on AWS Cloud.

The Environments realized are three:

* Test
* Preprod
* Production

# Cloud and Infrastructure Services

## Infrastructural Scope

Criticalcase boundary:

* Setup, configuration and tuning of all the infrastructure services:
  + [S3](https://aws.amazon.com/it/s3/)
  + [CodeBuild](https://aws.amazon.com/it/codebuild/)
  + [CodeDeploy](https://aws.amazon.com/it/codedeploy/)
  + [Codepipeline](https://aws.amazon.com/it/codepipeline/)
* Create all the Environments via Automation

## Architecture

## Codepipeline

### Pipeline deploy code

Each environment has 4 Codepipelines for deploy the code, 1 for each service within the servers:

|  |  |  |
| --- | --- | --- |
| Name | Scope | Environment |
| dolly-test-gcods-fcl | deploy for the fcl service | test |
| dolly-test-gcods-dashboard | deploy for the dashboard service | test |
| dolly-test-gcods-witech | deploy for the witech service | test |
| dolly-test-gcods-ignite | deploy for the ignite service | test |
| dolly-preprod-gcods-fcl | deploy for the fcl service | preprod |
| dolly-preprod-gcods-dashboard | deploy for the dashboard service | preprod |
| dolly-preprod-gcods-witech | deploy for the witech service | preprod |
| dolly-preprod-gcods-ignite | deploy for the ignite service | preprod |
| dolly-prod-gcods-fcl | deploy for the fcl service | prod |
| dolly-prod-gcods-dashboard | deploy for the dashboard service | prod |
| dolly-prod-gcods-witech | deploy for the witech service | prod |
| dolly-prod-gcods-ignite | deploy for the ignite service | prod |

Each pipeline retrieves the code from a zipped and versioned file stored in a bucket on S3. In the file, in addition to the code, there is also an appspec.yml document used by CodeDeploy to know where to place the code inside the machine and which services to restart:

1. version: 0.0
2. os: linux
3. files:
4. - source: /
5. destination: /usr/share/gcods/ignite/
6. file\_exists\_behavior: OVERWRITE
7. hooks:
8. AfterInstall:
9. - location: Scripts/restart\_services.sh

The S3 buckets in which the code for the deploy is loaded are the following:

|  |  |  |
| --- | --- | --- |
| Name | Scope | Environment |
| gcods.codepipeline.source.test | store the code to be deployed on the EC2 servers | test |
| gcods.codepipeline.source.preprod | store the code to be deployed on the EC2 servers | preprod |
| gcods.codepipeline.source.prod | store the code to be deployed on the EC2 servers | prod |

Within them, there is this sub-folder structure for each service:

|  |  |
| --- | --- |
| Name | Scope |
| build/build.zip | store the buildspec.yml file for create the ami |
| dashboard/dashboard.zip | store the code for the dashboard service |
| fcl/fcl.zip | store the code for the fcl service |
| ignite/ignite.zip | store the code for the ignite service |
| witech/witech.zip | store the code for the witech service |

### Pipeline create AMI

Each environment has 1 CodePipeline for create the AMI and refresh the autoscaling

|  |  |  |
| --- | --- | --- |
| Name | Scope | Environment |
| dolly-test-gcods-create-ami | create the ami that will be used in the autoscaling and start the instance refresh | test |
| dolly-preprod-gcods-create-ami | create the ami that will be used in the autoscaling and start the instance refresh | preprod |
| dolly-prod-gcods-create-ami | create the ami that will be used in the autoscaling and start the instance refresh | prod |

Each pipeline has the task of creating an AMI starting from the master servers outside the autoscaling. When it is ready, an instance refresh of the autoscaling of the corresponding environment will be performed. The code is present on S3 in a zipped file. Inside there is a buildspec.yml file used by CodeBuild to know which commands from aws cli to execute:

1. ---
2. phases:
3. build:
4. commands:
5. - "echo Build started on `date`"
6. - "date\_now=$(date '+%d-%m-%Y-%H-%M-%S')"
7. - "ami\_id=$(aws ec2 create-image --instance-id $instanceId --name $Name-$date\_now --no-reboot --output text)"
8. - "echo $ami\_id"
9. - "aws ec2 wait image-available --image-ids $ami\_id"
10. - "aws ec2 create-launch-template-version --launch-template-id $ltGcods --source-version '$Latest' --launch-template-data '{\"ImageId\":\"'$ami\_id'\"}'"
11. - "aws autoscaling start-instance-refresh --auto-scaling-group-name $Autoscaling --preferences '{\"SkipMatching\":true,\"MinHealthyPercentage\":0}'"
12. version: 0.2

Each CodeBuild has in its own variables the information necessary to run, such as the instance id of the master server, the name of the autoscaling, etc ...