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.

Criticalcase S.r.l.

P.IVA: 09733390018 | REA: TO - 1076960
CAP. SOC.: 120.000,00 Euro i.v.

SEDE LEGALE - Via Nicola Fabrizi, 136 - 10145 – Torino
SEDE OPERATIVA - Via Chambery 93/107, 10142 Torino
Tel: 011.5097366 | Fax: 011.04.32.771
info@criticalcase.com | criticalcase.com

Summary

1 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

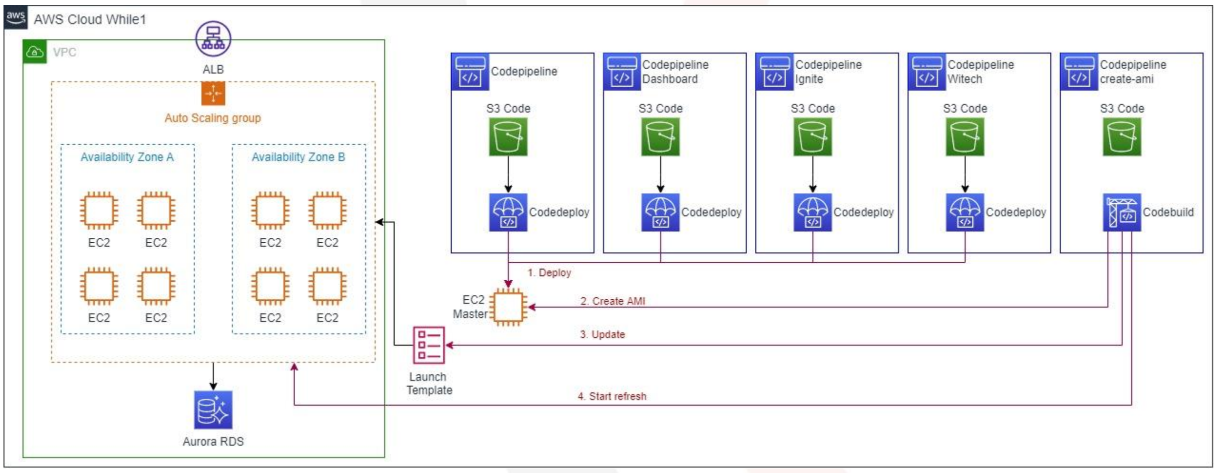
2 Cloud and Infrastructure Services

2.1 Infrastructural Scope

Criticalcase boundary:

- Setup, configuration and tuning of all the infrastructure services:
  - S3
  - CodeBuild
  - CodeDeploy
  - Codepipeline
- Create all the Environments via Automation

2.2 Architecture



2.3 Codepipeline

2.3.1 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:

```
version: 0.0
os: linux
files:
  - source: /
    destination: /usr/share/gcods/ignite/

file_exists_behavior: OVERWRITE

hooks:
  AfterInstall:
    - 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

2.3.2 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:

```
---
phases:
  build:
    commands:
      - "echo Build started on `date`"
      - "date_now=$(date '+%d-%m-%Y-%H-%M-%S')"
      - "ami_id=$(aws ec2 create-image --instance-id $instanceId --name $Name-$date_now --no-reboot --output text)"
      - "echo $ami_id"
      - "aws ec2 wait image-available --image-ids $ami_id"
      - "aws ec2 create-launch-template-version --launch-template-id $ltGcods --source-version '$Latest' --launch-template-data '{\"ImageId\":\"'$ami_id'\"}'"
      - "aws autoscaling start-instance-refresh --auto-scaling-group-name $Autoscaling --preferences '{\"SkipMatching\":true,\"MinHealthyPercentage\":0}'"
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 ...