International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 4 Issue 4, June 2020 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Building and Deploying a Static Application using Jenkins and Docker in AWS

Malathi. S[1], Ganeshan. M[2]

[1]PG Scholar, [2]Assistant Professor,

[1,2]Department of MCA, School of CS and IT, Jain (Deemed to be University), Bengaluru, Karnataka, India

ABSTRACT

Although there are many ways to deploy the Jenkins open-source automation server on Amazon Web Services (AWS), this whitepaper focuses on two specific approaches. First, the traditional deployment on top of Amazon Elastic Compute Cloud (Amazon EC2). Second, the containerized deployment that leverages Amazon EC2 Container Service (Amazon EBS). These approaches enable customers to take advantage of the continuous integration/ continuous delivery (CI/CD) capabilities of Jenkins. Using an extensive plugin system, Jenkins offers options for integrating with many AWS services and can morph to fit most use cases. Suppose you've built a new application for your client, or maybe yourself, and have managed to get a good user base that likes your application. You've gathered feedback from your users, and you go to your developers and ask them to build new features and make the application ready for deployment. With that ready, you can either stop the entire application and deploy the new version or build a zero downtime CI/CD deployment pipeline which would do all the tedious work of pushing a new release to users without manual intervention. we will talk exactly about the latter, how we can have a continuous deployment pipeline of a three-tier web application built in Node.js on AWS Cloud using Terraform as an infrastructure orchestrator. We'll be using Jenkins for the continuous deployment part and Bitbucket to host our codebase. we will look into setting up a Jenkins server which will be used for our CI/CD pipeline. We will be using Terraform and AWS for setting this up as well. The Terraform code for setting Jenkins is inside the folder Jenkins/setup. we have the AMIs for the API and web modules, we will trigger a build to run Terraform code for setting up the entire application and later go through the components in Terraform code which makes this pipeline deploy the changes with zero downtime of service. The first thing is that Terraform provides these lifecycle configuration blocks for resources within which you have an option create_before_destroy as a flag which literally means that Terraform should create a new resource of the same type before destroying the current resource.

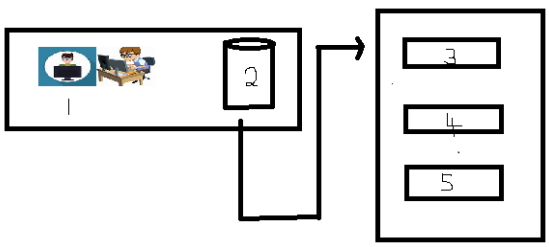
KEYWORDS: *Jenkins, AWS Cloud, web application built, configuration, CI/CD deployment pipeline, plugin system, Amazon EC2, Amazon EBS*

How to cite this paper: Malathi. S | Ganeshan. M "Building and Deploying a Static Application using Jenkins and Docker in AWS" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-4 | Issue-4, June 2020, pp.16-18, URL: www.ijtsrd.com/papers/ijtsrd30835.pdf

IJTSRD30835

INTRODUCTION

Jenkins: is an open source automation tool written in Java with plugins built for Continuous Integration purpose.[1] Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies. [3]Jenkins has over 2000 plugins integrated with other tools like docker, git, selenium etc. [2]By integrating with other tools it makes sure the software development is fully automated.



1. Developer, 2.Source Code Repository, 3.Builds, 4.Run Test, 5.Develops to a live server.
Fig: 1 JENKINS WORKING IMAGE

Now the Jenkins will pull the source code and builds, tests it and deploy it by using plugins and other tools. It can't only used for Continuous Integration also continuous Delivery, Continuous Deployment with the help of plugins.[1]By integrating with other tools the applications can be deployed to a testing environment. The user acceptance test and load

testing is performed to check the application is production ready this process is basically Continuous Delivery.[6] Now it can make use of plugins to continuous Deploy the application to a live server.

Continuous Delivery: [8]It is a process where you build software in such a way that it can be released to production at any time. Consider the diagram below:

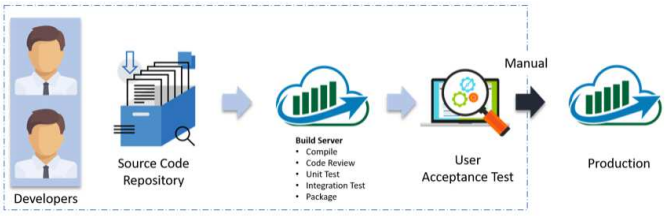


FIG: 1.2 CONTINUOUS DELIVERY

Automated build scripts will detect changes in Source Code Management (SCM) like Git. Once the change is detected, source code would be deployed to a dedicated build server to make sure build is not failing and all test classes and integration tests are running fine. [5]Then, the build application is deployed on the test servers (pre-production servers) for User Acceptance Test (UAT). Finally, the application is manually deployed on the production servers for release.

Docker: Running docker on aws provides developers and admins a highly reliable, low-cost way to build, ship, and run distributed applications at any scale. AWS supports both Docker licensing models: open source Docker community Edition (CE) and subscription-based Docker Enterprise Edition(EE).

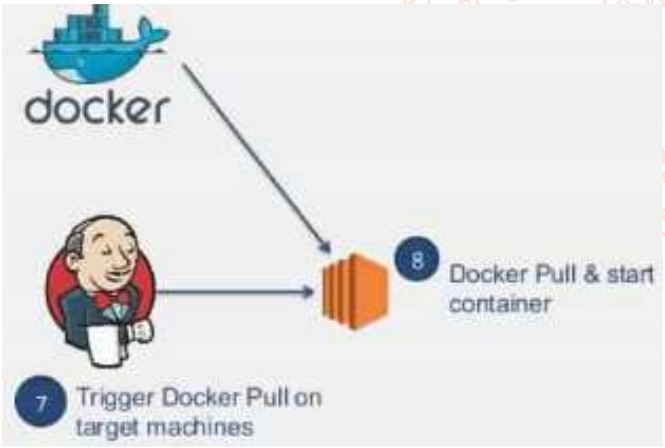


FIG: 1.2 DOCKER WORKING

Amazon Elastic Compute Cloud (Amazon EC2): Provides scalable computing capacity in the Amazon Web Services (AWS) cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

In this we are using general purpose T2 medium instances. We are pleased to announce the immediate availability of Amazon EC2 T2 instances. T2 instances are a new low-cost, General Purpose instance type that are designed to provide a baseline level of CPU performance with the ability to burst above the baseline. [8]With On-Demand Instance prices starting at $0.013 per hour ($9.50 per month), T2 instances are the lowest-cost Amazon EC2 instance option and are ideal for web servers, developer environments, and small databases.

T2 instances are for workloads that don't use the full CPU often or consistently, but occasionally need to burst to higher CPU performance.[3]Many applications such as web servers, developer environments and small databases don't need consistently high levels of CPU, but benefit significantly from having full access to very fast CPUs when they need them. T2 instances are engineered specifically for these use cases.

T2 instances are available in three sizes: t2.micro, t2.small, and t2.medium and work well in combination with Amazon EBS General Purpose (SSD) volumes for instance block storage.

Instance Type			
vCPU			
Memory (GiB)			
	t2.micro	t2.small	t2.medium
	1	1	2
	1	2	4

T2 instances are backed by the latest Intel Xeon processors with clock speeds up to 3.3 GHz during burst periods.

ECR: Amazon Elastic Container Registry (Amazon ECR) is a managed AWS Docker registry service that is secure, scalable, and reliable. Amazon ECR supports private Docker repositories with resource-based permissions using AWS IAM so that specific users or Amazon EC2 instances can access repositories and images. Developers can use the Docker CLI to push, pull, and manage images.

Cloud 9: AWS Cloud9 is a cloud-based integrated development environment (IDE) that lets you write, run, and debug your code with just a browser. It includes a code editor, debugger, and terminal. Cloud9 comes prepackaged with essential tools for popular programming languages, including JavaScript, Python, PHP, and more, so you don't need to install files or configure your development machine to start new projects. Since your Cloud9 IDE is cloud-based, you can work on your projects from your office, home, or anywhere using an internet-connected machine. [10]Cloud9 also provides a seamless experience for developing serverless applications enabling you to easily define resources, debug, and switch between local and remote execution of serverless applications. With Cloud9, you can quickly share your development environment with your team, enabling you to pair program and track each other's inputs in real time.

Ansible: Ansible is an infrastructure automation platform that makes it easy to manage and configure your servers. Vagrant allows us to create reproducible environments, making it really easy to work with virtual machines. We'll

use Ansible to automate the installation of Jenkins CI in a fresh CentOS image, created by Vagrant.

Requirements

Before we start, make sure you have vagrant and ansible installed. Installing it through homebrew should be really straightforward:

brew install ansible && ansible –version

brew cask install vagrant && vagrant -v

Gitlab: GitLab is a complete DevOps platform, delivered as a single application that provides everything you need to Manage, Plan, Create, Verify, Package, Release, Configure, Monitor, and Secure your applications. In this 18 minute recorded video demo, you will see GitLab's built-in issue tracking, issue boards, version control, code review, Auto DevOps, CI/CD, SAST/DAST and more.

Using all there tools we are building and deploying a static application in AWS.

PROPOSED ENHANCEMENT: Here I am using CI/CD pipelinest is latest one used in Jenkins. Continuous deployment smaller code changes are simpler more atomic. Fault isolation is simpler and quicker. Mean time to resolution is shorter because of the smaller code changes and quicker fault isolation.

EXISTING SYSTEM:

It is there in manually. Manual testing is a type where testers manually execute test cases without using any automation tools.

ARCHITECTURAL DIAGRAM OF THE PROJECT

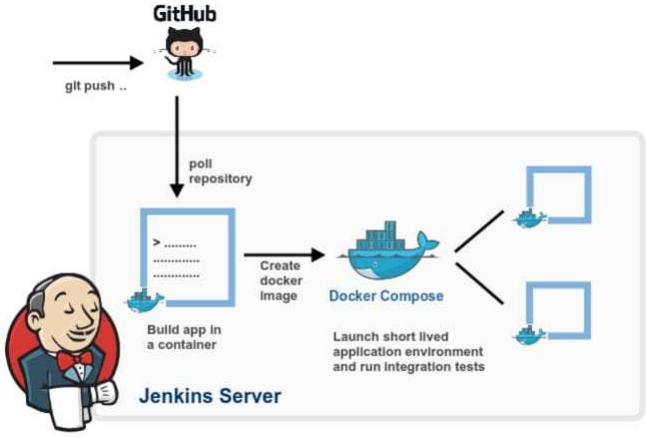


FIG: 2.1 ARCHITETURE DIAGRAM

DATA FLOW DIAGRAM

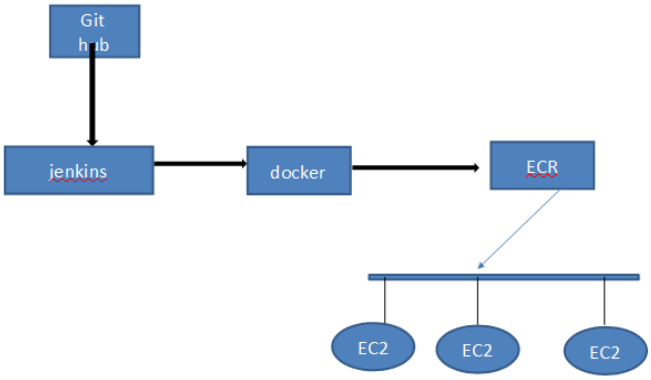


FIG: 2.2 DATAFLOW DIGRAM

CONCLUTION:

Here we are the conclution of our article. Integrating Jenkins is almost seamless with any existing project-life cycle due to the abundant library of plugins and free documentation all over the internet. What we've seen here is just a small portion of what Jenkins has to offer as a CI tool. In this article, we focused mostly on Jenkins as a CI tool, we haven't change our application code much except the docker file update to accommodate the test results. The continuous integration with a docker article covers the addition of integration tests in grater detail. The entire projects available under docker series GITHUB repository under the docker-series-continuous integration-Jenkins-end branch.

REFERENCES:

[1] https://tutcris.tut.fi/portal/files/17171909/2018_CNAX_Workshop_Aleksi_8.pdf

[2] https://d0.awsstatic.com/whitepapers/aws-building-fault-tolerant-applications.pdf

[3] https://hostadvice.com/how-to/how-to-use-docker-containers-with-aws-ec2/

[4] https://docs.aws.amazon.com/AmazonECR/latest/userguide/images.html

[5] https://aws.amazon.com/blogs/devops/set-up-a-build-pipeline-with-jenkins-and-amazon-ecs/

[6] End to End Automation on Cloud with Build Pipeline: The Case for DevOps in Insurance Industry, Continuous Integration, Continuous Testing, and Continuous Delivery

[7] Continuous Integration, Continuous Testing, and Continuous Delivery Mitesh Soni IGATE Gandhinagar, Indiamitesh.soni@igate.com

[8] Building a virtually air-gapped secure environment in AWS: with principles of devops security program and secure software delivery

[9] https://link.springer.com/chapter/10.1007/978-3-030-03673-7_4

[10] Implementation of a Dev Ops Pipeline for Server less Applications