MICROLAND®
Extraordinary. Everyday.

WHITEPAPER

Go Agile: The Docker Way

Abstract

Continuous Integration (CI) and Continuous Delivery (CD) practices have emerged as the standard for modern software testing and delivery. The Docker solution accelerates and optimizes CI/CD pipelines, while maintaining a clear separation of concerns between the development and operations teams. Integrating Docker into the CI pipeline has helped many organizations accelerate system provisioning, reduce job time, increase the volume of jobs run, enable flexibility in language stacks and improve overall infrastructure utilization.

With Docker containers, you can define your own runtime environment. You can choose your own platform, programming language, and any application dependencies (such as package managers or tools), that are not supported by other platforms. Docker containers are self-contained and include all the configuration information and software your web application requires to run.

In this paper, we discuss the construction of a CI/CD pipeline that makes use of Docker along with various other technologies. The pipeline is kicked off by a code commit to a GitHub repository and runs through a complete work flow to deploy the code to test.

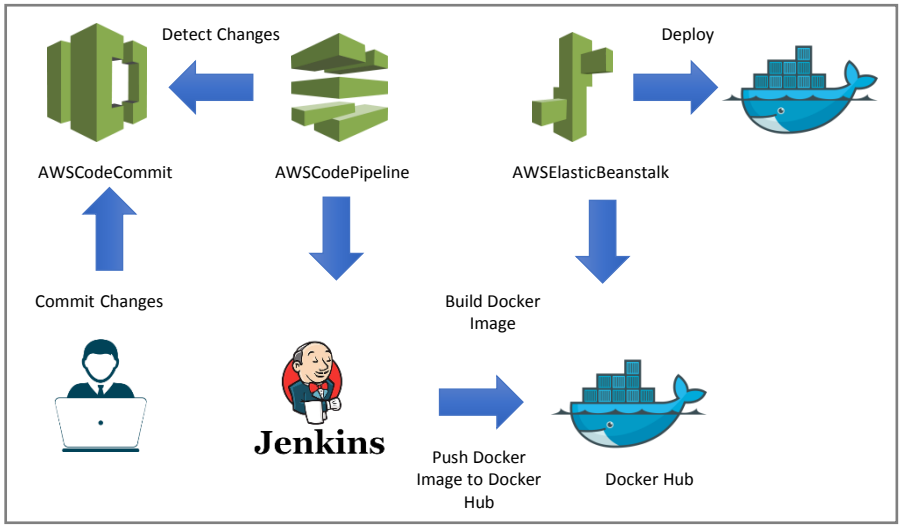
WHITEPAPER

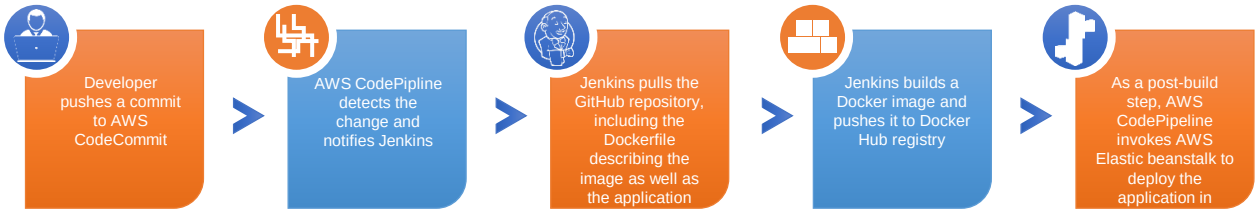
Go Agile: The Docker Way

Solution

Components

AWS CodeCommit
AWS CodePipeline
Jenkins
Docker Hub
Docker Engine
AWS Elastic beanstalk





Commit of the code on GitHub (CodeCommit) automatically triggers Jenkins to run a Docker image build job. On successful completion of the build job, Jenkins pushes a Docker image up to Docker Hub.

Elastic Beanstalk supports the deployment of web applications from Docker containers.
By using Docker with Elastic Beanstalk, you have an infrastructure that automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring. You can manage your web application in an environment that supports the range of services that are integrated with Elastic Beanstalk, including, but not limited to, VPC, RDS, and IAM.

The CodeCommit repository should have the code of the application to be tested along with Dockerfile to build the application node, and any code necessary for testing. Finally, the Jenkins master takes advantage of the AWS CodePipeline plugin, which allows a Jenkins job to be triggered when a change is pushed to a CodeCommit repository.

The immutability of Docker images ensures a repeatable deployment with what is developed, tested through CI, and run in production. Docker Engine deployed on Elastic beanstalk allows the containers to be portable across environments. Docker registry allows DevOps and release teams to manage container images in a single location through the entire release process.

Environment Setup

Prerequisites

The base installation and configuration of the various components are not covered in this solution. It is assumed before beginning the steps outlined in this document that the following prerequisites have been met:
Installed Jenkins Master server with the following components

- AWS CodePipeline plugin
- Maven
- Docker engine for building Docker image
- Docker Hub account
- Configured AWS account with the following services/components
- AWS CodeCommit
- AWS CodePipeline
- AWS Elastic Beanstalk
- API Keys

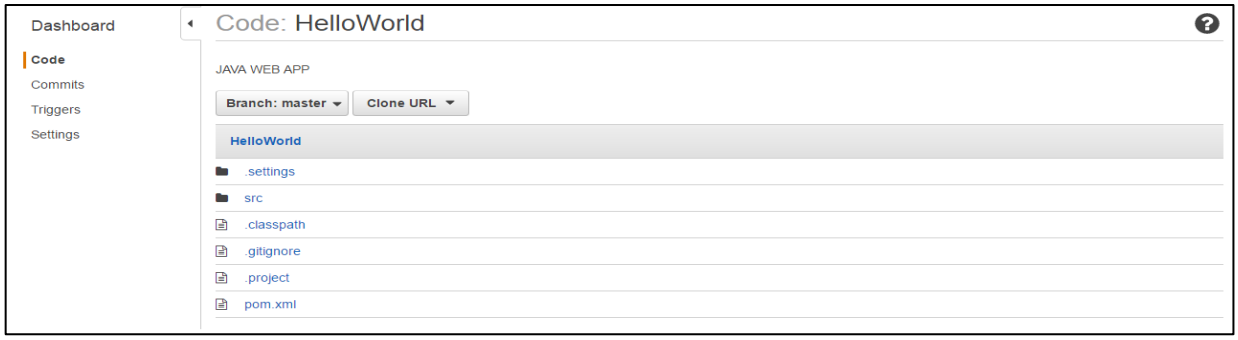
All components can communicate with each other on the network, including domain name resolution and all the necessary certificates have been installed.

Configuring AWS CodeCommit

In order to create and test a target application, the CodeCommit repository for the target application needs to contain the following components:

The application code

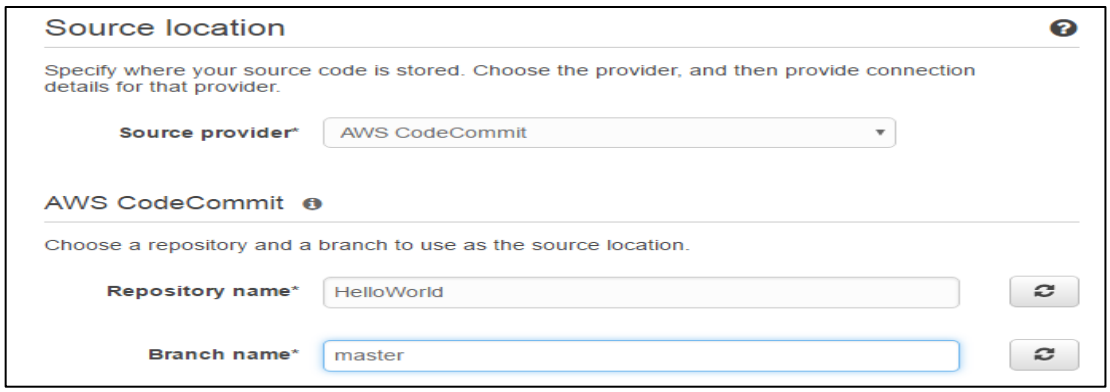
A Dockerfile that describes how to build the application container, and copies over the necessary application and test code.
As an example, the following image depicts the repository for a Hello World Java application:



Note: This Dockerfile is presented as an example; there is nothing in the file specific to enabling the CI workflow. The only requirement for this workflow is that there is a Dockerfile that will build the requisite image needed for testing. In addition to the presence of a Dockerfile, the repository needs to be configured to notify the Jenkins server when a new commit happens. This is done via CodePipeline.

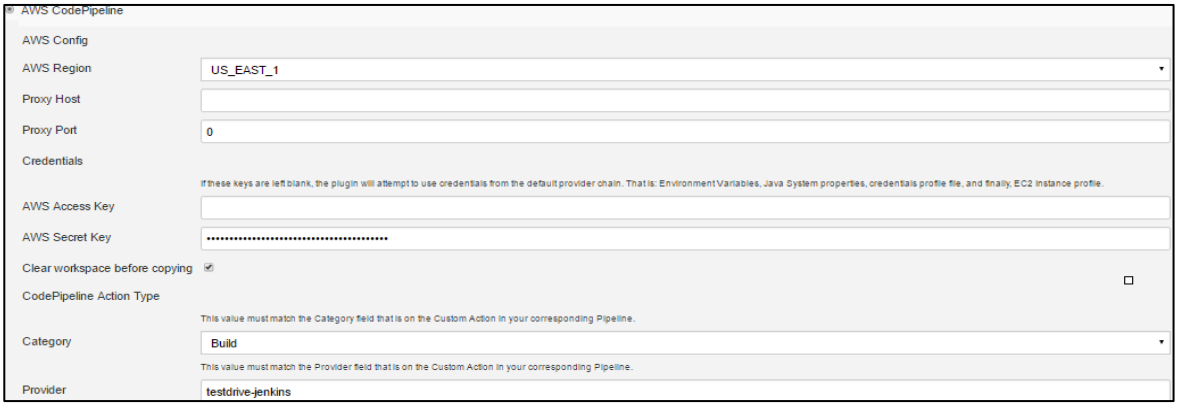
The Dockerfile is constructed as follows:

```
FROM preetick/jvcatenv:v1

MAINTAINER M1017042

# Location of presentation URL
ADD  /HelloWorldMaven.war /webapps/

ADD start_up.sh /

CMD bash start_up.sh
```



Configuring the Jenkins Master

After the base Jenkins image is installed and the service is up and running, AWS CodePipeline plugin needs to be installed on the Jenkins master. This plugin initiates a Jenkins job when a change is pushed to a designated repository. In addition to the AWS CodePipeline plugin, Docker build and publish plugin also needs to be installed on the Jenkins master in order to build Docker images and publish them to Docker Hub.



Configuring AWS Elastic Beanstalk

Elastic Beanstalk supports the deployment of web applications from Docker containers.

Single Container Docker

The single container configuration can be used to deploy a Docker image (described in a Dockerfile or Dockerrun.aws.json definition) and source code to EC2 instances running in an Elastic Beanstalk environment. Use the single container configuration when you only need to run one container per instance.

Steps to launch an environment with a sample application (console)

1.Open the Elastic Beanstalk console.

2.Choose an application or create a new one.

3.In the upper right corner, choose Create New Environment.

4.Choose Create web server.

5.For Predefined configuration, choose Docker and Single instance.

6.Choose Next.

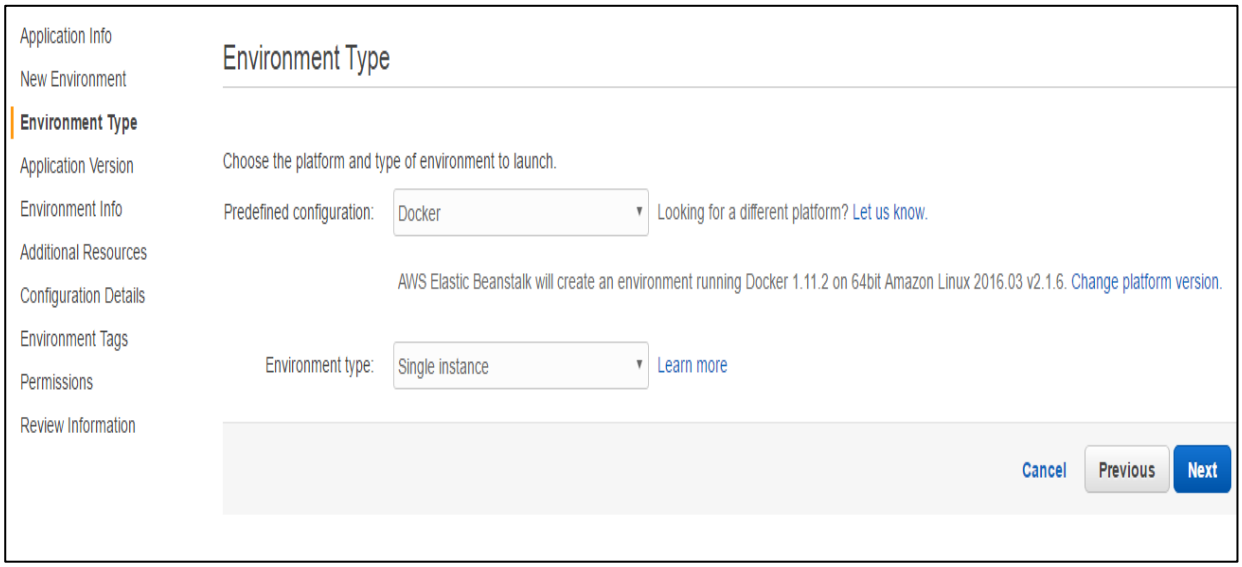
7.For Application Version, choose Sample Application under Existing application version, or choose Upload your own and upload an application source bundle. Choose Next.

8.Type an Environment name and URL prefix for your environment, and then choose Next.

9.If the application requires a database, choose Create an RDS DB Instance.

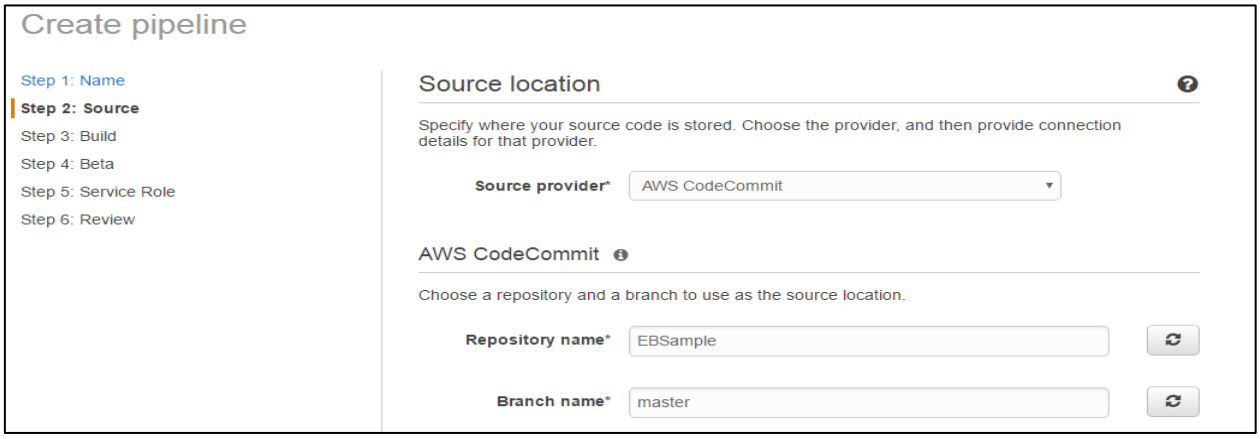
10.Choose Next.

11.For Configuration Details, set Instance type to t2.micro, optionally select an EC2 key pair, and then choose Next. Assigning a key pair allows you to connect to instances in your environment for debugging.

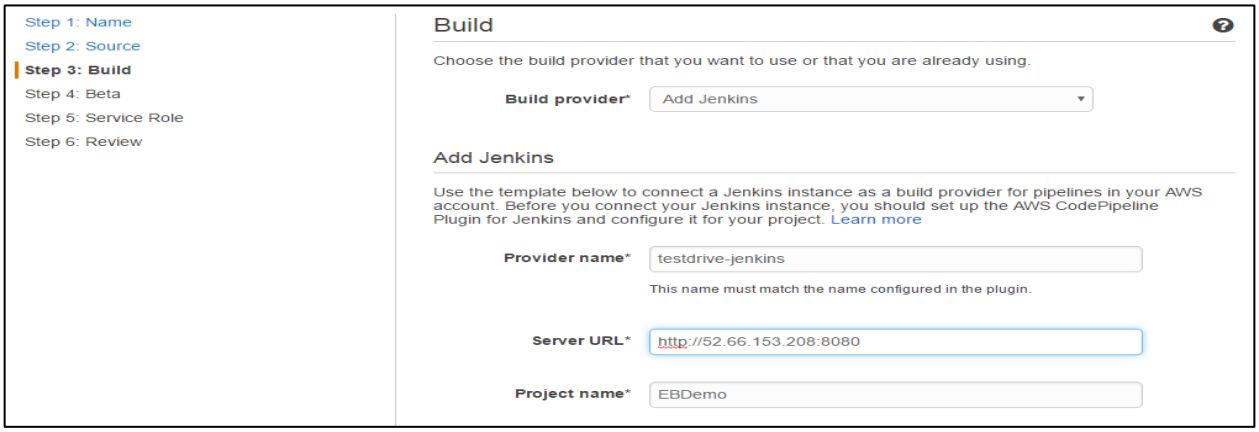


Configuring AWS CodePipeline

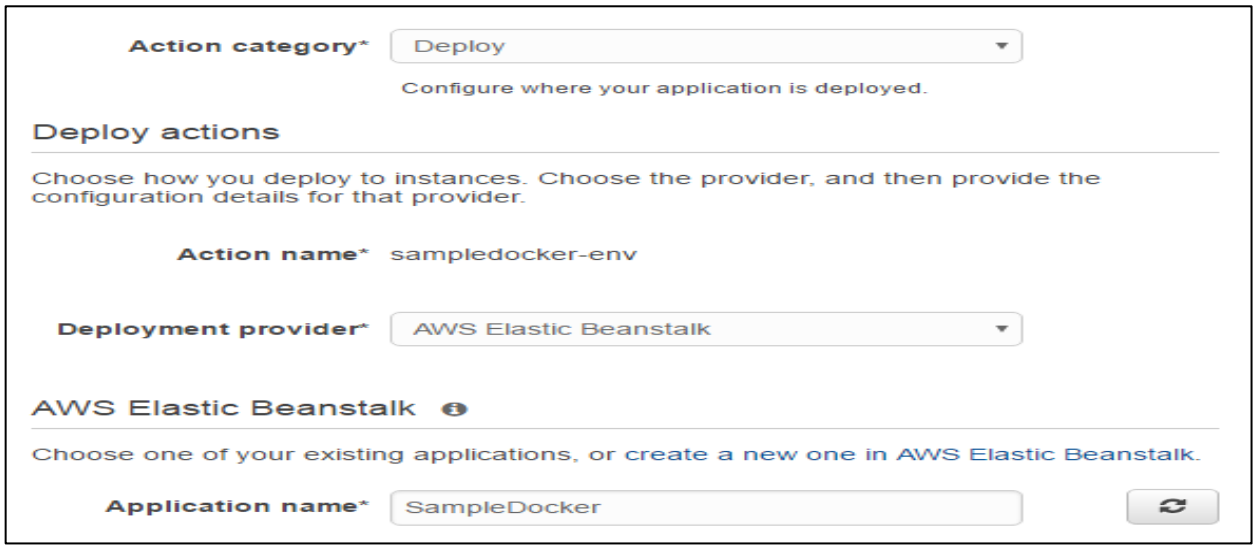
1. Create a new pipeline by selecting CodeCommit as the source provider for debugging.



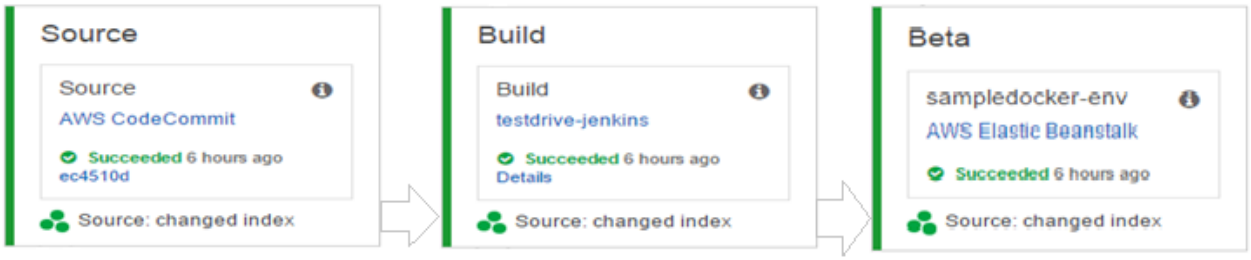
2. Select Jenkins as the build provider



3. Select AWS Elastic Beanstalk as the deploy service



4. The pipeline will look like this



Putting it all together: Running a test

To kick off the workflow, a developer makes a commit to the applications CodeCommit repository. This triggers CodeCommit webhook, which notifies Jenkins to execute the appropriate tests.

Jenkins receives the webhook and builds a Docker image based on the Dockerfile contained in the CodeCommit repo. After the image is built, it is pushed to Docker Hub. CodePipeline then invokes Elastic Beanstalk and deploys the application code to a Docker container.

Conclusion

Organizations that leverage Docker as part of their continuous integration pipeline can expect to increase stability and agility, and reduce complexity of their software development processes. Docker allows users to ensure consistent repeatable processes, while simultaneously enabling a reduction in the number of test images that need to be tracked and maintained. The lightweight nature of containers means they can be spun up faster, driving more rapid test cycles.

WHITEPAPER

Go Agile: The Docker Way

The Microland Solution

Microland's DevOps services facilitate adoption of the 'automate-and-version-control-everything' approach to drive integration and automation of toolsets, self-service portals, and SLA-driven services (remote and on-site).

What you can expect

Microland's cloud DevOps services deliver the following outcomes:

Maximize cloud investments
Through DevOps automation framework, continuous integration, cloud orchestration, and configuration tools, we maximize ROI of your DevOps environments

Cost optimization
smartCenter and smartGovern frameworks enable cost-effective management of multi-tool and multi-cloud DevOps environments

Faster turnaround
Employing DevOps best practices helps increase productivity and resilience

What sets us apart
Microland has vast expertise in building DevOps environments on public and private clouds using open source and cloud native tools. Our differentiators include:

Proprietary toolset
smartGovern identifies cost optimization opportunities and enforces organization-specific governance and compliance requirements on AWS infrastructure
smartDevOps automates the complete development-to-release cycle

Reach us at **cloud@microland.com** for further information

References

Amazon VPC Network Connectivity Options–
http://media.amazonwebservices.com/AWS_Amazon_VPC_Connectivity_Options.pdf
Amazon Elastic Beanstalk-
http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html
Amazon CodeCommit-
http://docs.aws.amazon.com/codecommit/latest/userguide/welcome.html
Amazon CodePipeline-
http://docs.aws.amazon.com/codepipeline/latest/userguide/welcome.html
Docker-
https://www.docker.com/what-docker

About the authors

Rahul Natarajan is a Cloud Solutions Architect with experience developing technical partnerships, managing enterprise projects, leading technical teams, and architecting cloud-based solutions. He currently works at Microland as a Lead Architect – DC and Cloud Practice. He helps customers architect, optimize, and manage enterprise solutions running on the AWS IaaS/PaaS platform, and develop AWS architectures to solve key technical challenges around DevOps, complex cloud migrations, scalability, high availability and disaster recovery. He has led successful projects with large enterprises, start-ups, finance and education customers globally.

WHITEPAPER

Go Agile: The Docker Way

For further information

Contact us at cloud@micoland.com

About Microland

Microland is a leading Hybrid IT Infrastructure Service Provider and a trusted partner to enterprises in their IT-¬as-a-Service journey. Incorporated in 1989 and headquartered in Bangalore, India, Microland has more than 3,400 professionals across its offices in Europe, Middle East, North America and India. Microland enables global enterprises to become more agile and innovative through a comprehensive portfolio of services that addresses hybrid IT transformation, workspace transformation, service transformation and end-to-end IT infrastructure management.

Lear more about us at:

www.microland.com

Microland Limited
1B, Ecospace, Bellandur Outer Ring Road, Bangalore 560 103, India
P: +91 80 3918 0000 | **F:** +91 80 3918 0044 | www.microland.com

MICROLAND®
Extraordinary. Everyday.