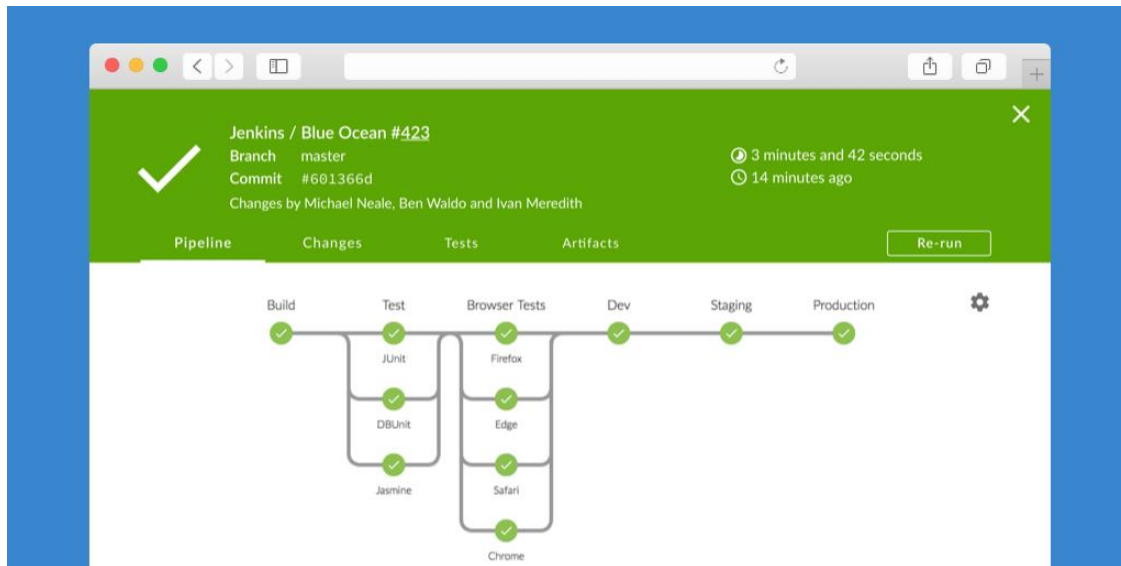


There are several ways you can install Jenkins in your master node (EC2 instance in our case).

Using Docker – the best because it established the blue ocean pipeline



Blue Ocean, a continuous delivery tool designed for Jenkins, makes up for those limitations by letting users create, visualize, and monitor Jenkins-based CI/CD pipelines through a graphical interface. This can help alleviate the labor-intensive process of manually building pipelines in Jenkins.

Using Yum

Using a War file

I would prefer the 1st method, using docker which comes with blue ocean plugin preinstalled and this will be used in rest of the Jenkins blog series.

<https://cloudaffaire.com/how-to-install-jenkins-in-aws-ec2-instance/>

Customers running microservices-based applications on **Amazon Elastic Kubernetes Service (Amazon EKS)** are looking for guidance on architecting complete end-to-end Continuous Integration (CI) and

Continuous Deployment / Delivery (CD) pipelines using Jenkins and Spinnaker.

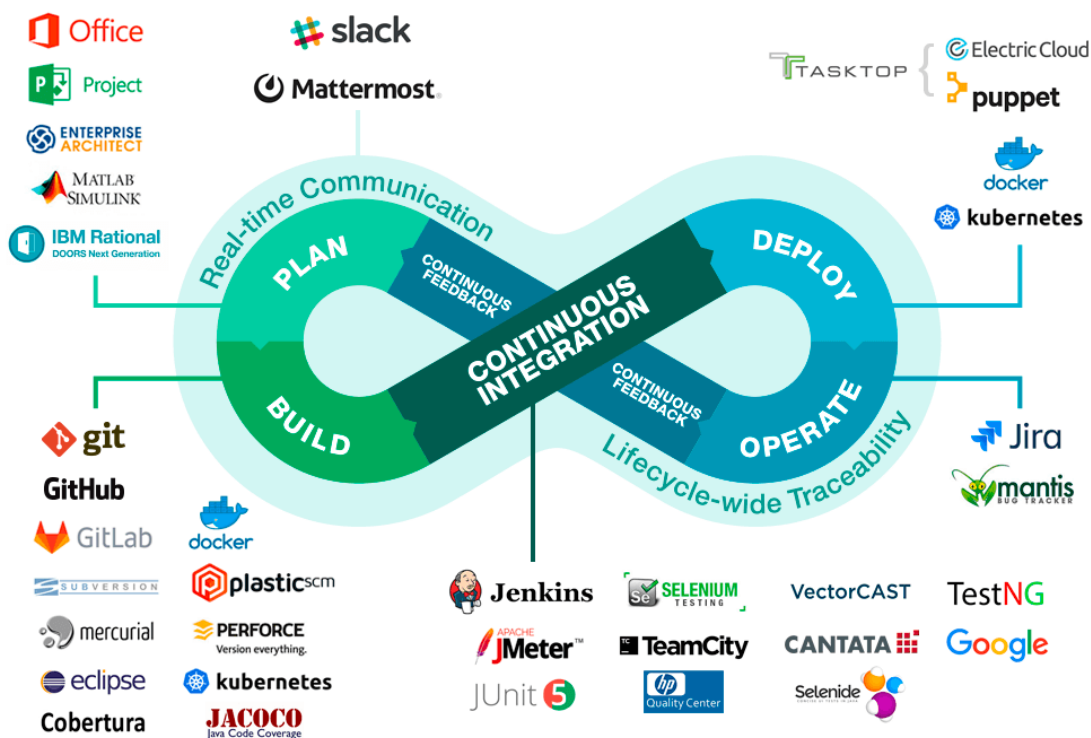
An Amazon EKS cluster consists of two primary components:

- The Amazon EKS control plane
- Amazon EKS nodes that are registered with the control plane

Jenkins is a very popular **CI server** with great community support and many plugins (Slack, GitHub, Docker, Build Pipeline) available.

Slack is a collaboration software that supports users in communicating, correlate, and cooperate efficiently within and among units.

Trello, JIRA is a project management software that allows many users to maintain and prioritize tasks and modules.



Spinnaker provides automated release, built-in deployment, and supports **blue/green deployment out of the box**.

Spinnaker is an open-source multi-cloud continuous delivery platform for releasing software changes with high velocity and confidence. It provides two core sets of features: **cluster management** and **deployment management**.

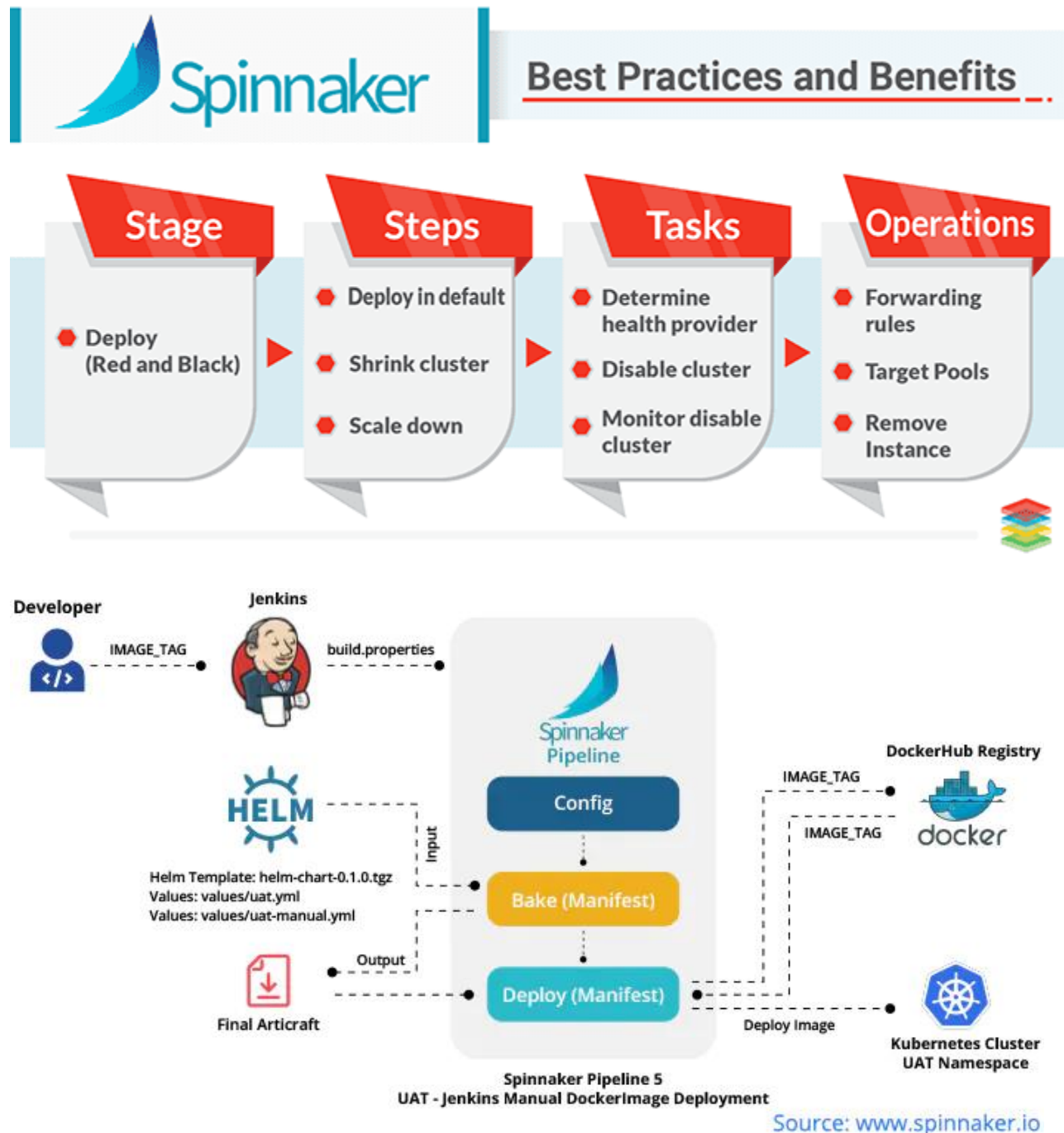
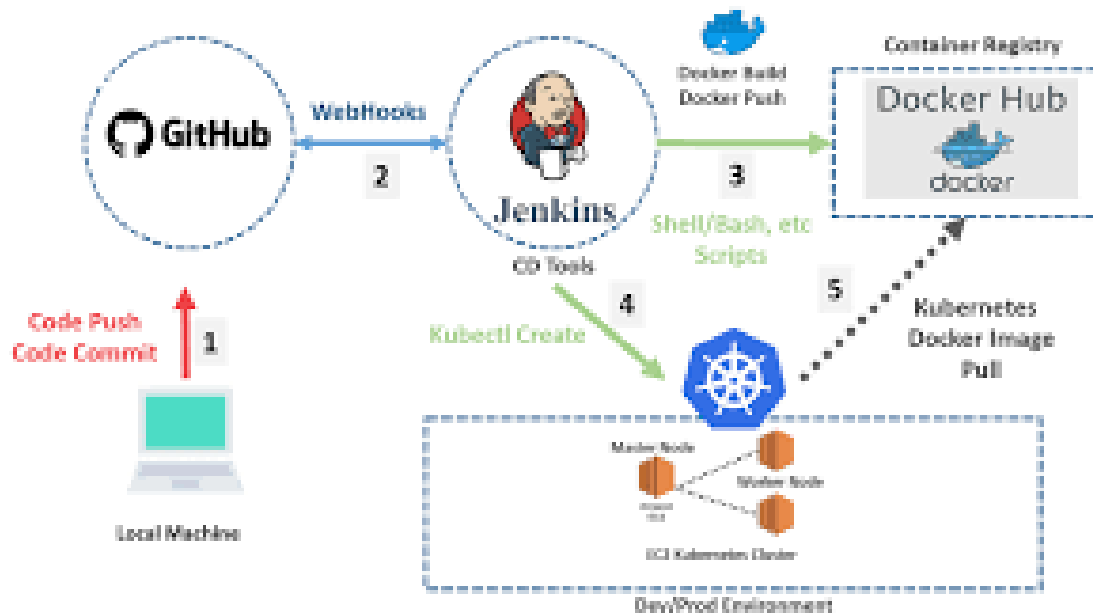
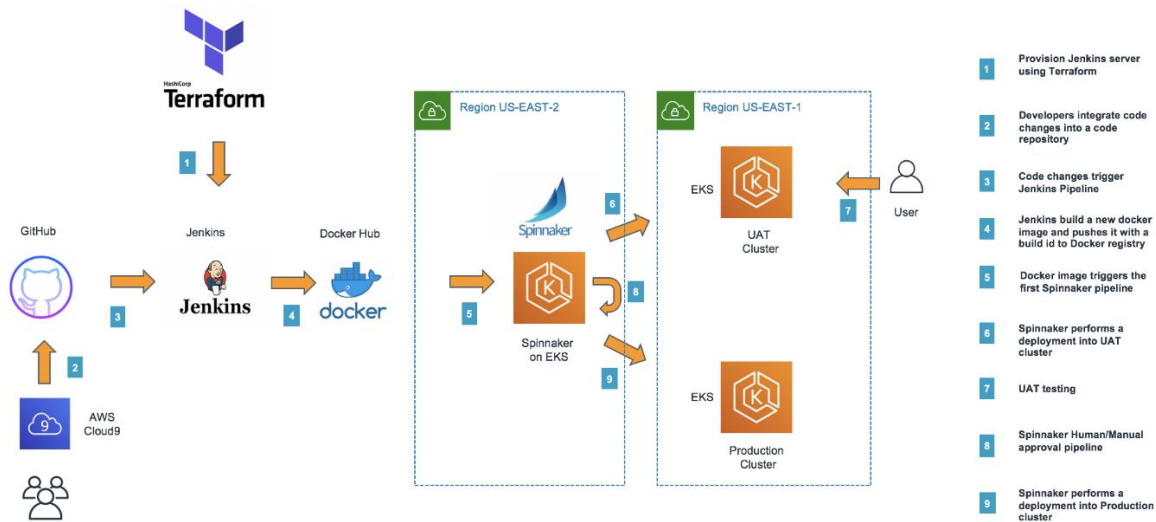


Image 5: UAT - Jenkins Manual Docker Image Deployment Pipeline Architecture



Practice on Kubernetes cluster/linode

Manifest

Type of deployment strategies

- Canary
- Rollingupdate
- Bluegreen

We are going to provision Jenkins using terraform

Overview of concepts

Terraform

Resume points for terraform

Terraform is a tool for building/provisioning, changing, and versioning infrastructure safely and efficiently.

It is controlled via an easy-to use command line interface (CLI), as well as a free-to-use **SaaS** offering called Terraform Cloud and a private installation for enterprise. (terraform cloud)

Terraform can manage existing and popular service providers as well as custom in-house solutions.

Terraform cloud to store the state file, access providers and manage the access control of the state file.

Configuration files describe to Terraform the components needed to run a single application or your entire datacenter.

Terraform generates an execution plan describing what it will do to reach the desired state, and then executes it to build the described infrastructure.

As the configuration changes, Terraform is able to determine what changed and create incremental execution plans which can be applied.

The key features of Terraform are: **Infrastructure as Code**, Execution Plans, Resource Graph, and Change Automation.

Troubleshooting of different scenarios connected to the configuration files whenever terraform return an error message during the init, plan, apply stages.

Jenkins

Jenkins is a self-contained, open source automation server which can be used to automate **all sorts of tasks related to building, testing, and delivering or deploying software**. It can be installed through native system packages, **Docker (the best because we have a blue ocean container)**, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.

Prerequisites

To implement the instructions in this post, you will need the following:

- AWS account
- Docker Hub account
- GitHub account
- How to manually create a keyname on aws

git clone <https://github.com/joshking1/amazon-eks-jenkins-terraform.git>

Go to my github account and fork this repo

```
cd amazon-eks-jenkins-terraform/terraform/
```

```
cd terraform/
```

```
terraform init
```

```
terraform plan
```

```
terraform apply -auto-approve
```

```
terraform apply -auto-destroy
```

Terraform apply will also output the IP address of the Jenkins CI server as shown above.

When you create a keyname, save in the same folder used to clone the forked repo.

Let visit the way to install Jenkins and challenges you might face

Daemon issue on aws – solve with `sudo amazon-linux-extras install epel -y`

Terraform will provision an AWS EC2 instance and install git, Apache Maven, Docker, Java 8, and Jenkins as shown in the **install_jenkins.sh** file:

```
#!/bin/bash
```

```
sudo yum -y update
```

```
echo "Install Java JDK 8"
```

```
sudo yum remove -y java
```

```
sudo yum install -y java-1.8.0-openjdk
```

```
echo "Install Maven"
```

```
sudo yum install -y maven
```

```
echo "Install git"
```

```
sudo yum install -y git
```

```
echo "Install Docker engine"
```

```
sudo yum update -y
```

```
sudo yum install docker -y
```

```
sudo sudo chkconfig docker on
```

```
echo "Install Jenkins"
```

```
sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo
```

```
sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key
```

```
sudo yum install -y Jenkins
```

```
sudo amazon-linux-extras install epel -y
```

```
sudo usermod -a -G docker jenkins
```

```
sudo chkconfig jenkins on
```

```
echo "Start Docker & Jenkins services"
```

```
sudo service docker start
```

```
sudo service jenkins start
```

Using a browser, open the page at http://jenkins_ip_address:8080; the Jenkins admin page will be displayed:

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

Fig 3. Jenkins admin page

Using the AWS Cloud9 shell terminal, log in to the Jenkins CI server, find the Administrator password by running the following command:






```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Enter this Administrator password on the Jenkins Console by pasting it into the input box, and click Next. Click Install suggested plugin.

Configure Jenkins

1. Plugins:

Log in to the Jenkins console, click Manage Jenkins → Manage Plugins → Available. Choose and install Docker plugin and GitHub Integration Plugin, then restart Jenkins by clicking the Restart Jenkins check box as shown here:

Docker API	 Downloaded Successfully. Will be activated during the next boot
Docker	 Downloaded Successfully. Will be activated during the next boot
Icon Shim	 Downloaded Successfully. Will be activated during the next boot
GitHub Integration	 Downloaded Successfully. Will be activated during the next boot

 [Go back to the top page](#)
(you can start using the installed plugins right away)

 ☐ Restart Jenkins when installation is complete and no jobs are running

Fig 4. Jenkins plugins

2. Credentials:

Docker Hub: Click Credentials → global → Add Credentials, choose Username with password as Kind, enter the Docker Hub username and password and use **dockerHubCredentials** for ID.

GitHub: Click Credentials → Global → Add Credentials , choose Username with password as Kind, enter the GitHub username and password and use **gitHubCredentials** for ID.