**Knowledge Transfer Document**

**Topics:**

* Jenkins Development
* Deployment Configuration and Security checklist
* Automation Deployment [ JIRA - JENKINS ].
* Automation Application Deployments.
* OKE Jenkins Configurations.
* Docker installation and guidance.
* Monitoring Tool - Installation and configurations ( Grafanna, Prometheus ).
* Lambda for Automate the Billing Report.

**Jenkins Development:**

* Implemented the Secrets instead of direct username and password into the Jenkins from Secret Manager using following Plugins.

1. AWS Secrets Manager Credentials Provider
2. AWS Secrets Manager SecretSource



* Created the separate users in Jenkins using [Matrix Authorization Strategy Plugin](https://plugins.jenkins.io/matrix-auth" \t "http://10.11.22.38:8080/manage/pluginManager/_blank). Use Role based User.

1. Create a role and assign the roles to the users.



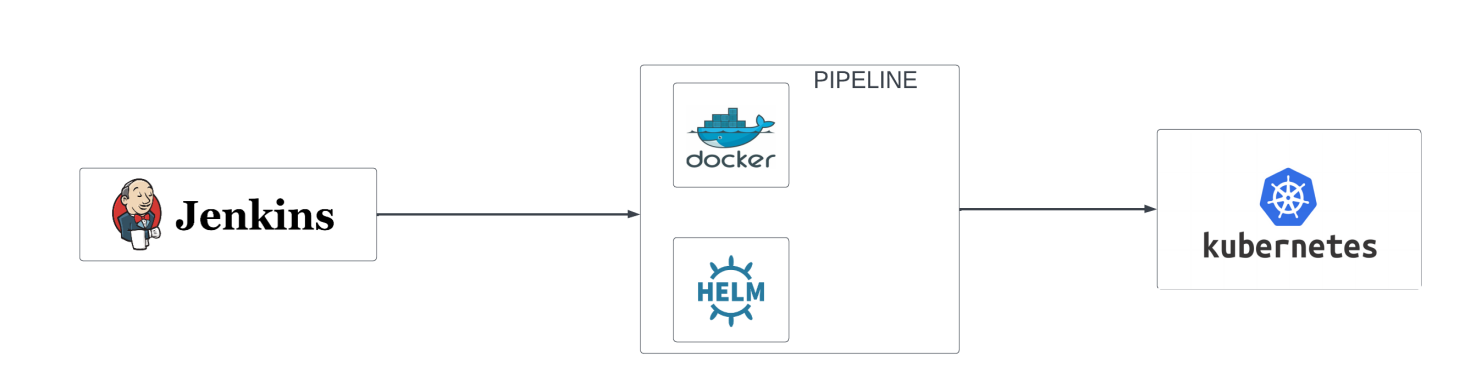
* Configure the token based user for JIRA JENKINS Connections.
* Attached the logs in the Jira deployment tickets for the deployment updates.
* Deployment status checks are configured in the JENKINS Pipeline.
* Modified the pods size based upon the performance and application need, on both Production and UAT.
* Created Job in Jenkins to kill 10 days old images in docker to avoid the memory issue. JobName : [rmnpdockerimage](http://10.11.22.38:8080/job/rmnpdockerimage/)



Installation Guide :

Reference Link : [https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-22-04]("Installation)

**Deployment Configuration:**



* Created a branch for Helm setup and change the deployment and service files accordingly. Please have a look to this below repo FYI.

**<https://git-codecommit.ap-south-1.amazonaws.com/v1/repos/helm>**

* Implemented the WAF and Tags in the loadbalancer ( Ingress JSON file).

Note : BU and APP-NAME is compulsory.

**Secrets Key Creation for DB :**

* Creating the secret key for the databases ( RDS and MONGODB ). Follow the below link for creation.

Ref Link : <https://docs.aws.amazon.com/secretsmanager/latest/userguide/create_database_secret.html>

* Note : If we need to create a Secret key for Replica Database, have to create a JSON and run the AWS CLI for creation. Refer the below JSON.

{

"username": "uname\*\*\*\*",

"password": "pword\*\*\*\*",

"engine": "mysql | mongo",

"host": "dbhostname\*\*\*\*\*",

"port": 4406,

"dbInstanceIdentifier": "identifier\*\*\*\*"

}

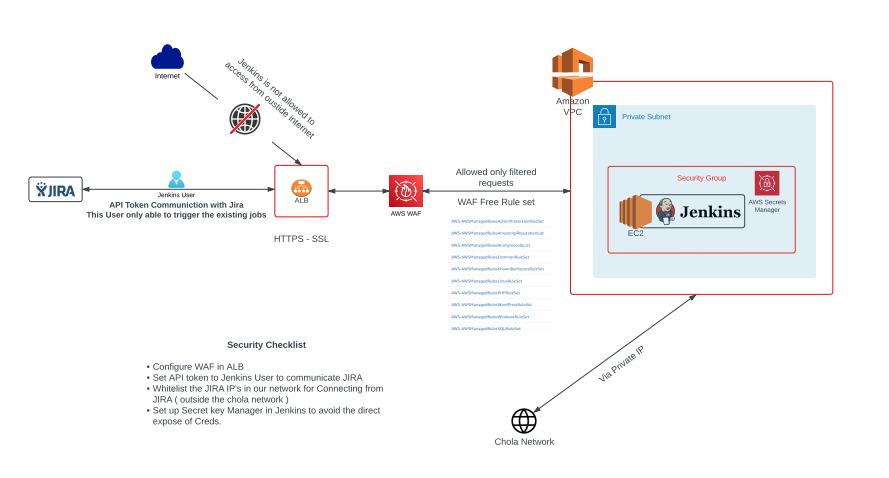
* Create a JSON file and run the AWS CLI with the filename.

**Example**

Filename : **rds\_secret.json**

AWS CLI : “**aws secretsmanager create-secret --name MyTestSecret --secret-string file://rds\_secret.json”**

**Jenkins Security Improvements** :

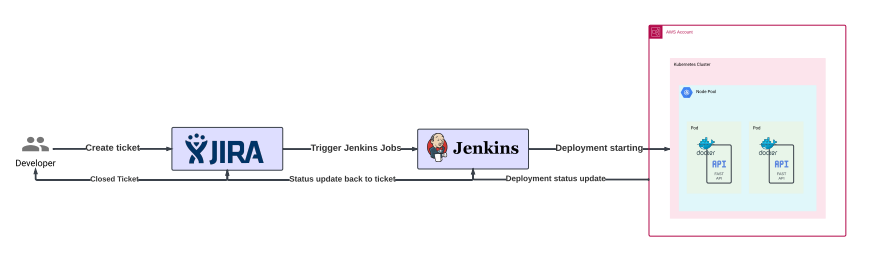


* Configured AWS Secret Manager for Code Commit Integration into the Jenkins.
* Secured all plain text and key value secrets from AWS Secret Manager.
* Removed the unwanted and vulnerable Plugins.
* Configured the well defined port instead of default port.
* All users are protected by the Strong Password policy as follows.
  + **Password minimum length** Minimum password length of 8 characters and a maximum length of 128 characters.
  + **Password strength** Minimum of three of the following mix of character types: uppercase, lowercase, numbers, and non-alphanumeric character.

Suggestions:

* Adding all users to Active Directory for better security for login and integrate the MFA.
* Mail Alert - when there is a new job is created or if the jobs taking too long.

**Automation Deployment [ JIRA - JENKINS ] :**



* Create a Jira web hooks and configure it in the Jenkins.
* Configured the Jenkins pipeline to get the deployment status back to the JIRA Ticket.
* Integrate the Jenkins pipeline with the cluster via the pipeline to get the application deployment status.

**Docker installation and guidance.**

Ref link :

<https://docs.docker.com/engine/install/ubuntu/>

* Install the latest version of Docker using the above link.
* Configure the ECR Repo’s into the Jenkins server to pull and push the Containers. It is necessary to configure our UAT Environment - ECR.

**CMD** : *aws ecr get-login-password | docker login -u AWS --password-stdin "https://$(aws sts get-caller-identity --query 'Account' --output text).dkr.ecr.$(aws configure get region).amazonaws.com" ---* to login into the AWS Container.

* Once the login is finished, we can build and push our build into ECR.

**Note :**

* + - In case of Docker storage Issue - please look for the **/var/lib/docker/overlay2** directory.
    - Unmount the directory and restart the docker

**Lambda for Automate the Billing Report :**

Ref link:

<https://aws.amazon.com/blogs/architecture/email-delta-cost-usage-report-in-a-multi-account-organization-using-aws-lambda/>

* This is used to create a report for all AWS Accounts and send the mail with the respective team.
* We deployed this lambda function in the **Payer Account** to access all other account billing services using **BOTO3 Python.**
* All the program related are present in the above link.

**Monitoring Installation and configurations ( Grafanna, Prometheus ).**

* Grafanna used as the monitoring tool and Prometheus is used as log metrics that passes to grafanna for detailed monitoring.
* Follow the above link to configure the Grafanna and Prometheus.

**Reference link :**

<https://medium.com/techlogs/grafana-loki-with-aws-s3-backend-through-irsa-in-aws-kubernetes-cluster-93577dc482a>

<https://www.guru99.com/create-users-manage-permissions.html>