Technical Achievements

1. Ansible Automation
2. Openshift Install
3. Infrastructure layer
4. Linux task/ Installation
5. Docker and Containers
6. Cache Database on AIX
7. Epic Clarity/ Caboodle(ETL and Dataware House) setup with tools
8. MLOPs Training
9. AWS Cloud Learnings and setup
10. SQL Server Database Configurations and Designing
11. cloudpak

# Ansible Automation

Have created Multiple playbooks to automate the tasks in Linux

1. Created users, directories, files on multiple machines using ansible
2. Installation and Upgradation of packages
3. Configured a web server using Ansible
4. Configured multiple playbooks using variables, conditions, distributions, handler blocks
5. Created Ansible Galaxy/ Ansible Roles

# Openshift Installation

1. Used static IP configuration For Openshift Installation on Vmware Vpshere
2. Created the Virtual machines with the configurations required to setup Openshift on top of it.

**Openshift 4.2 Release Notes:**

In OpenShift Container Platform 4.2, you require access to the internet to install and entitle your cluster.

OpenShift Container Platform 4.2 is supported on Red Hat Enterprise Linux 7.6 and later, as well as Red Hat Enterprise Linux CoreOS 4.1.

Must use Red Hat Enterprise Linux CoreOS (RHCOS) for the control plane, or master, machines and can use either RHCOS or Red Hat Enterprise Linux 7.6 for compute, or worker, machines.

A picture containing screenshot

Description automatically generated

**Cluster Deployment Topology**

**Multiple Masters Using Native HA with Co-located Clustered etcd : Openshift 4.**

|  |  |
| --- | --- |
| Host Name | Component/Role(s) to Install |
| Installer Node + DHCP+ Http | Jump Host all Deployment node user for upgrade, install cluster |
| Boot Strap Node | The bootstrap machine boots and starts hosting the remote resources required for control plane machines to boot. |
| master1.example.com | Master (clustered using native HA) and node with etcd running as a static pod on each host |
| master2.example.com | Master (clustered using native HA) and node with etcd running as a static pod on each host |
| master3.example.com | Master (clustered using native HA) and node with etcd running as a static pod on each host |
| lb.example.com | External Load Balancer |
| node1.example.com | Compute node |
| node2.example.com | Compute node |
| Node3.example.com | Compute node |
| Node4.example.com | Compute node |
| infra-node1.example.com | Dedicated infrastructure node |
| infra-node2.example.com | Dedicated infrastructure node |
| Infra-node3.example.com | Dedicated infrastructure node |
| Infra-node4.example.com | Dedicated infrastructure node ( Optional Node ) |

**Sizing System Requirement: Red Hat Enterprise Linux CoreOS (RHCOS)**

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Machine | Operating System | No of Node | vCPU | RAM ( GB) | Storage |
| Installer Node | RHEL 8 | 1 | 8 | 8 | 1X1000G |
| Bootstrap | RHCOS | 1 | 16 | 16 | 1X1000G |
| Master Control plane | RHCOS | 3 | 3X8 | 3X32 | 3X1000G |
| Compute | RHCOS | 4 | 4X8 | 4X32 | 4X1000G |

OCP 4.2 on VMWare with RHCOS Deployment Flow

A screenshot of a cell phone

Description automatically generated

1. Logging and Monitoring

OpenShift Container Platform uses Prometheus, Alert manager and Grafana for monitoring the platform and deployed applications.

Monitoring components are deployed by default as part of OCP installation, however there is no option to specify the persistent volume name as part of install. Hence, the persistent volume claim will wait for the required persistent volumes and need to configure alert for Deployment components

Required Persistent Volume for E-NAS

Here are the required persistent volumes: Size we can change as per our requirements

2 x 50 GB volumes for Prometheus

3 x 10 GB volumes for alertmanager

# Infrastructure layer

The below diagram shows the high level architecture of the components involved in the setup of the project

![A screenshot of a video game

Description automatically generated]()

# Linux task/ Installation

User creation, access management, ssh configuration

Network configuration using nmcli

Packages Installation, upgradation

Scheduling jobs

Partitioning

Automation of multiple tasks using ansible

Documentation of LLD and as builds

Performed Installation of RHEL 8

# Docker and Containers

1. Pulled the images from docker hub, updated with some required packages and further created the image out of it which is used for cloning other docker containers.
2. Hands on to share the OS using – Export Import using file share, SCP and Pushing Image into Registry(hub.docker.com)
3. Networking in Docker- SNATTING, DENATTING, PATTING
4. Load Balancing- HA Proxy and DNS

# Cache Database on AIX

1. Caché is designed to transcend the limitations of the relational model while providing an evolutionary upgrade path for the thousands of existing relational database applications as well as support for the many SQL-based reporting tools on the market.

In addition to being a high-performance object database, Caché is also a full-featured relational database. All the data within a Caché database is available as true relational tables and can be queried and modified using standard SQL via ODBC, JDBC, or object methods. Because of the power of the underlying Caché database engine, we believe that Caché is the fastest, most reliable, and most scalable relational database available today.

1. Configured Cache along with EPIC to handle DHA’s data. Epic cache was configured on top of IBM AIX
2. I was the secondary resource in order to handle this big data and been trained on the same. Support and maintain InterSystems Caché databases for Epic systems application environments in accordance with established policies and procedures.  
   Install, upgrade, and support the various Epic Cache database application environments using transparent upgrade and RA/SU installs.

# Epic Clarity/ Caboodle(ETL and Dataware House) setup with tools

The Clarity Process

The Console and the Clarity Process

The Console is a web-based interface that provides centralized management tools for all aspects of the Clarity data transfer process. It serves as a communication box between the Caché environment and Clarity server. From the Console, you can define the properties of the Clarity process, set up execution files, schedule the execution files to run, and check on the status of running or finished execution files.

The Console facilitates management of the Clarity process from a centralized user interface while remaining flexible enough to handle issues that occur during the transfer process and to accommodate various workflows.

![A close up of a device

Description automatically generated]()

Here Chronicles store the main database which further be extracted by Clarity Console tool and further sent to SQL Server in order to do reporting.

Been involved in all the configuration and setup of the tool and extracted the data using several methods and provided further for the reporting.

Caboodle(Datawarehouse)

Further extraction of data from clarity happens to caboodle along with the external data which can be used in further data analysis.

# Machine Learning and AI

Taking personal training on MLOPS and learning python to create the programs for machine learning and AI

Created the program to take the image like cheese

Program to crop the image

Face Recognition

Computer vision

Image Processing

Written above programs using python Jupyter Notebook

# AWS Cloud Learnings and setup

Created multiple templates on Linux and windows platform.

Storage and Network configuration for the same.

Deployed the VM’s using existing templates.

Also understand the concept of public and private cloud along with Hybrid cloud.

Worked on below AWS Services and Categories-

Compute

• EC2

• Elastic Container Service (ECS)

• Elastic Beanstalk

• AWS Batch

• AWS Lambda

Storage

• Elastic Block Store (EBS)

• Simple Storage Service (S3)

• Glacier

• Amazon Elastic File System

• AWS Storage Gateway

• AWS Snowball

Database

• Amazon Aurora

• Amazon RDS

• Amazon DynamoDB

• Amazon ElastiCache

• Amazon Redshift (Data

warehouse)

• Amazon Neptune (Graph

database)

• AWS Database Migration Service

Networking

• Amazon VPC

• Amazon CloudFront

• Amazon Route 53

• Amazon API Gateway

• AWS Direct Connect

• Elastic Load Balancing

# SQL Server Database Configurations and Designing

Designed several databases as per the client requirement and configured the same.

Performed the DR setup and did the successful failover for the same.

Database Clustering, Replication, always-on and configured same on client environment.

Administer & Support mission critical databases and providing on-call Production support.

Created the Virtual Machines for the same and configured the roles required for the same.