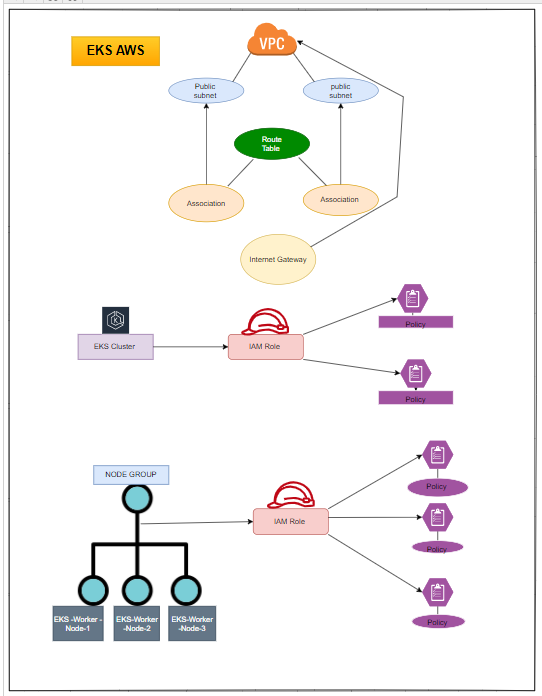
**Paywallet**

**Environment configuration Guide**

1. **Creating EKS in AWS**
   1. **Architecture diagram of EKS cluster environment**

**\*\*\* All the below details are as of today.**



**Configuration Details:**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

**Node details:**

\*\*\* All the below details as of today.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generatedGraphical user interface, application

Description automatically generated

For EKS cluster we are using the creating the resource group as follows

Region chosen for creating all resources: us-east-1

**1.2. Terraform Scripts for EKS cluster.**

<https://github.com/Maveric-Digital/devops-config/tree/master/Terraform-AWS>

Text

Description automatically generated

**eks-cluster.tf**

|  |
| --- |
| #  # EKS Cluster Resources  # \* IAM Role to allow EKS service to manage other AWS services  # \* EC2 Security Group to allow networking traffic with EKS cluster  # \* EKS Cluster  #  resource "aws\_iam\_role" "paywallet-cluster" {  name = "terraform-eks-paywallet-cluster"  assume\_role\_policy = <<POLICY  {  "Version": "2012-10-17",  "Statement": [  {  "Effect": "Allow",  "Principal": {  "Service": "eks.amazonaws.com"  },  "Action": "sts:AssumeRole"  }  ]  }  POLICY  }  resource "aws\_iam\_role\_policy\_attachment" "paywallet-cluster-AmazonEKSClusterPolicy" {  policy\_arn = "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"  role = aws\_iam\_role.paywallet-cluster.name  }  resource "aws\_iam\_role\_policy\_attachment" "paywallet-cluster-AmazonEKSVPCResourceController" {  policy\_arn = "arn:aws:iam::aws:policy/AmazonEKSVPCResourceController"  role = aws\_iam\_role.paywallet-cluster.name  }  resource "aws\_security\_group" "paywallet-cluster" {  name = "terraform-eks-paywallet-cluster"  description = "Cluster communication with worker nodes"  vpc\_id = aws\_vpc.paywallet.id  egress {  from\_port = 0  to\_port = 0  protocol = "-1"  cidr\_blocks = ["0.0.0.0/0"]  }  tags = {  Name = "terraform-eks-paywallet"  }  }  resource "aws\_security\_group\_rule" "paywallet-cluster-ingress-workstation-https" {  cidr\_blocks = [local.workstation-external-cidr]  description = "Allow workstation to communicate with the cluster API Server"  from\_port = 443  protocol = "tcp"  security\_group\_id = aws\_security\_group.paywallet-cluster.id  to\_port = 443  type = "ingress"  }  resource "aws\_eks\_cluster" "paywallet" {  name = var.cluster-name  role\_arn = aws\_iam\_role.paywallet-cluster.arn  vpc\_config {  security\_group\_ids = [aws\_security\_group.paywallet-cluster.id]  subnet\_ids = aws\_subnet.paywallet[\*].id  }  depends\_on = [  aws\_iam\_role\_policy\_attachment.paywallet-cluster-AmazonEKSClusterPolicy,  aws\_iam\_role\_policy\_attachment.paywallet-cluster-AmazonEKSVPCResourceController,  ]  } |

**eks-worker-nodes.tf**

|  |
| --- |
| #  # EKS Worker Nodes Resources  # \* IAM role allowing Kubernetes actions to access other AWS services  # \* EKS Node Group to launch worker nodes  #  resource "aws\_iam\_role" "paywallet-node" {  name = "terraform-eks-paywallet-node"  assume\_role\_policy = <<POLICY  {  "Version": "2012-10-17",  "Statement": [  {  "Effect": "Allow",  "Principal": {  "Service": "ec2.amazonaws.com"  },  "Action": "sts:AssumeRole"  }  ]  }  POLICY  }  resource "aws\_iam\_role\_policy\_attachment" "paywallet-node-AmazonEKSWorkerNodePolicy" {  policy\_arn = "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"  role = aws\_iam\_role.paywallet-node.name  }  resource "aws\_iam\_role\_policy\_attachment" "paywallet-node-AmazonEKS\_CNI\_Policy" {  policy\_arn = "arn:aws:iam::aws:policy/AmazonEKS\_CNI\_Policy"  role = aws\_iam\_role.paywallet-node.name  }  resource "aws\_iam\_role\_policy\_attachment" "paywallet-node-AmazonEC2ContainerRegistryReadOnly" {  policy\_arn = "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"  role = aws\_iam\_role.paywallet-node.name  }  resource "aws\_eks\_node\_group" "paywallet" {  cluster\_name = aws\_eks\_cluster.paywallet.name  node\_group\_name = "paywallet"  node\_role\_arn = aws\_iam\_role.paywallet-node.arn  subnet\_ids = aws\_subnet.paywallet[\*].id  scaling\_config {  desired\_size = 1  max\_size = 2  min\_size = 1  }  depends\_on = [  aws\_iam\_role\_policy\_attachment.paywallet-node-AmazonEKSWorkerNodePolicy,  aws\_iam\_role\_policy\_attachment.paywallet-node-AmazonEKS\_CNI\_Policy,  aws\_iam\_role\_policy\_attachment.paywallet-node-AmazonEC2ContainerRegistryReadOnly,  ]  } |

**providers.tf**

|  |
| --- |
| terraform {  required\_version = ">= 0.12"  }  provider "aws" {  region = var.aws\_region  }  data "aws\_availability\_zones" "available" {}  # Not required: currently used in conjunction with using  # icanhazip.com to determine local workstation external IP  # to open EC2 Security Group access to the Kubernetes cluster.  # See workstation-external-ip.tf for additional information.  provider "http" {} |

**variables.tf**

|  |
| --- |
| variable "aws\_region" {  default = "us-east-1"  }  variable "cluster-name" {  default = "eks-paywallet"  type = string  } |

**vpc.tf**

|  |
| --- |
| #  # VPC Resources  # \* VPC  # \* Subnets  # \* Internet Gateway  # \* Route Table  #  resource "aws\_vpc" "paywallet" {  cidr\_block = "10.0.0.0/16"  tags = tomap({  "Name" = "terraform-eks-paywallet-node",  "kubernetes.io/cluster/${var.cluster-name}" = "shared",  })  }  resource "aws\_subnet" "paywallet" {  count = 2  availability\_zone = data.aws\_availability\_zones.available.names[count.index]  cidr\_block = "10.0.${count.index}.0/24"  map\_public\_ip\_on\_launch = true  vpc\_id = aws\_vpc.paywallet.id  tags = tomap({  "Name" = "terraform-eks-paywallet-node",  "kubernetes.io/cluster/${var.cluster-name}" = "shared",  })  }  resource "aws\_internet\_gateway" "paywallet" {  vpc\_id = aws\_vpc.paywallet.id  tags = {  Name = "terraform-eks-paywallet"  }  }  resource "aws\_route\_table" "paywallet" {  vpc\_id = aws\_vpc.paywallet.id  route {  cidr\_block = "0.0.0.0/0"  gateway\_id = aws\_internet\_gateway.paywallet.id  }  }  resource "aws\_route\_table\_association" "paywallet" {  count = 2  subnet\_id = aws\_subnet.paywallet.\*.id[count.index]  route\_table\_id = aws\_route\_table.paywallet.id  } |

**workstation-external-ip.tf**

|  |
| --- |
| #  # Workstation External IP  #  # This configuration is not required and is  # only provided as an example to easily fetch  # the external IP of your local workstation to  # configure inbound EC2 Security Group access  # to the Kubernetes cluster.  #  data "http" "workstation-external-ip" {  url = "http://ipv4.icanhazip.com"  }  # Override with variable or hardcoded value if necessary  locals {  workstation-external-cidr = "${chomp(data.http.workstation-external-ip.body)}/32"  } |

**outputs.tf**

|  |
| --- |
| #  # Outputs  #  locals {  config\_map\_aws\_auth = <<CONFIGMAPAWSAUTH  apiVersion: v1  kind: ConfigMap  metadata:  name: aws-auth  namespace: kube-system  data:  mapRoles: |  - rolearn: ${aws\_iam\_role.paywallet-node.arn}  username: system:node:{{EC2PrivateDNSName}}  groups:  - system:bootstrappers  - system:nodes  CONFIGMAPAWSAUTH  kubeconfig = <<KUBECONFIG  apiVersion: v1  clusters:  - cluster:  server: ${aws\_eks\_cluster.paywallet.endpoint}  certificate-authority-data: ${aws\_eks\_cluster.paywallet.certificate\_authority[0].data}  name: kubernetes  contexts:  - context:  cluster: kubernetes  user: aws  name: aws  current-context: aws  kind: Config  preferences: {}  users:  - name: aws  user:  exec:  apiVersion: client.authentication.k8s.io/v1alpha1  command: aws-iam-authenticator  args:  - "token"  - "-i"  - "${var.cluster-name}"  KUBECONFIG  }  output "config\_map\_aws\_auth" {  value = local.config\_map\_aws\_auth  }  output "kubeconfig" {  value = local.kubeconfig  } |

* 1. **To Login to AWS cluster:**

1. **Install the AWS CLI**

<https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html>

1. **Configure the Access key and Secret access key**

* **export AWS\_ACCESS\_KEY\_ID=***<Generate the access key for each user>*
* **export AWS\_SECRET\_ACCESS\_KEY=***<Generate the secret access key for each user>*
* **export AWS\_DEFAULT\_REGION=***us-east-1*

1. **Run the below command to login to the cluster.**

**“**aws eks --region us-east-1 update-kubeconfig --name eks-paywallet”

**Note: All the configuration is done in Bastion server. Login to bastion server to access the cluster.**

Graphical user interface, text, application, email

Description automatically generated

1. **Architecture Diagram of Dev-Environment in AWS**

Graphical user interface, application

Description automatically generated

**2.1 Containerization**

**Dockerfile and docker-compose for each service:**

For Backend Service: To run the micro-services in containers with Docker, first clone the repository from Github and build it (mvn clean install -DskipTests). Now write a Dockerfile to copy the resultant war file into container and run it using java –jar command. Place the Dockerfile under root directory of your project.

For Frontend Service: As the frontend is ReactJS application. Build the npm package on the on-prem server and copy the resultant build to docker image.

**2.1.1: luther-sales-salesrep-app**

**Dev,UAT and Prod:**

**Dockerfile:**

|  |
| --- |
| FROM node:12.2.0-alpine  RUN mkdir /app  WORKDIR /app  COPY package.json ./  RUN npm install  COPY . /app  RUN npm run build  EXPOSE 3010  CMD ["npm", "start"] |

**docker-compose.yml:**

|  |
| --- |
| version: "3.6"    networks:  frontend-net:    services:    luther-sales-salesrep-app:  image: luther-sales-salesrep-app  container\_name: luther-sales-salesrep-app\_container  build:  context: ./  dockerfile: Dockerfile  ports:  - "3010:3000"  restart: always  networks:  - frontend-net |

**2.1.2: luther-sales-borrower-app**

**Dev,UAT and Prod:**

**Dockerfile:**

|  |
| --- |
| FROM node:12.2.0-alpine  RUN mkdir /app  WORKDIR /app  COPY package.json ./  RUN npm install  COPY . /app  RUN npm run build  EXPOSE 3020  CMD ["npm", "start"] |

**docker-compose.yml:**

|  |
| --- |
| version: "3.6"    networks:  frontend-net:    services:    luther-sales-borrower-app:  image: luther-sales-borrower-app  container\_name: luther-sales-borrower-app\_container  build:  context: ./  dockerfile: Dockerfile  ports:  - "3020:3000"  restart: always  networks:  - frontend-net |

**2.1.3: order-management-service**

**Dev and UAT and docker-compose.yml:**

|  |
| --- |
| version: "3.6"    networks:  backend-net:  volumes:  backend-service-logs:    services:    order-management-service:  image: order-management-service  container\_name: order-management-service\_container  build:  context: .  dockerfile: Dockerfile  ports:  - "9010:9010"  environment:  MONGO\_INITDB\_DATABASE: luthersales  MONGO\_INITDB\_ROOT\_USERNAME: user  networks:  - backend-net  volumes:  - /data/backend-service-logs:/log |

**Dev Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/ordermanagement-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9010  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**UAT Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/ordermanagement-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9010  CMD ["java","-Dspring.profiles.active=uat","-jar","app.jar"] |

**Prod Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/ordermanagement-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9010  CMD ["java","-Dspring.profiles.active=prod","-jar","app.jar"] |

**2.1.4: otp-service**

**Dev and UAT and docker-compose.yml:**

|  |
| --- |
| version: "3.6"  networks:  backend-net:  volumes:  backend-service-logs:    services:  otp-service:  image: otp-service  container\_name: otp-service\_container  build:  context: .  dockerfile: Dockerfile  ports:  - "9030:9030"  environment:  MONGO\_INITDB\_DATABASE: luthersales  MONGO\_INITDB\_ROOT\_USERNAME: user  networks:  backend-net:  volumes:  - /data/backend-service-logs:/log |

**Dev Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/otp-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9030  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**UAT Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/otp-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9030  CMD ["java","-Dspring.profiles.active=uat","-jar","app.jar"] |

**Prod Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/otp-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9030  CMD ["java","-Dspring.profiles.active=prod","-jar","app.jar"] |

**2.1.5: user-management-service**

**Dev and UAT and docker-compose.yml:**

|  |
| --- |
| version: "3.6"    networks:  backend-net:  volumes:  backend-service-logs:    services:    user-management-service:  image: user-management-service  container\_name: user-management-service\_container  build:  context: .  dockerfile: Dockerfile  ports:  - "9060:9060"  environment:  MONGO\_INITDB\_DATABASE: luthersales  MONGO\_INITDB\_ROOT\_USERNAME: user  networks:  - backend-net  volumes:  - /data/backend-service-logs:/log |

**Dev Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/user-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9060  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**UAT Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/user-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9060  CMD ["java","-Dspring.profiles.active=uat","-jar","app.jar"] |

**Prod Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/user-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9060  CMD ["java","-Dspring.profiles.active=prod","-jar","app.jar"] |

**2.1.6: accounts-management-service**

**Dev and UAT and docker-compose.yml:**

|  |
| --- |
| version: "3.6"    networks:  backend-net:  volumes:  backend-service-logs:    services:  accounts-management-service:  image: accounts-management-service  container\_name: accounts-management-service\_container  build:  context: .  dockerfile: Dockerfile  ports:  - "9050:9050"  environment:  MONGO\_INITDB\_DATABASE: luthersales  MONGO\_INITDB\_ROOT\_USERNAME: user  networks:  - backend-net  volumes:  - /data/backend-service-logs:/log |

**Dev Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/accountmanagement-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9050  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**UAT Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/accountmanagement-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9050  CMD ["java","-Dspring.profiles.active=uat","-jar","app.jar"] |

**Prod Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/accountmanagement-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9050  CMD ["java","-Dspring.profiles.active=prod","-jar","app.jar"] |

**2.1.7: transaction-management-service**

**Dev and UAT and docker-compose.yml:**

|  |
| --- |
| version: "3.6"    networks:  backend-net:  volumes:  backend-service-logs:    services:  transaction-management-service:  image: transaction-management-service  container\_name: transaction-management-service\_container  build:  context: .  dockerfile: Dockerfile  ports:  - "9100:9100"  environment:  MONGO\_INITDB\_DATABASE: luthersales  MONGO\_INITDB\_ROOT\_USERNAME: user  networks:  - backend-net  volumes:  - /data/backend-service-logs:/log |

**Dev Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/transactionmanagement-service-1.0-SNAPSHOT.jar app.jar  EXPOSE 9100  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**UAT Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/transactionmanagement-service-1.0-SNAPSHOT.jar app.jar  EXPOSE 9100  CMD ["java","-Dspring.profiles.active=uat","-jar","app.jar"] |

**Prod Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/transactionmanagement-service-1.0-SNAPSHOT.jar app.jar  EXPOSE 9100  CMD ["java","-Dspring.profiles.active=prod","-jar","app.jar"] |

**2.1.8: payroll-provider-service**

**Dev and UAT and docker-compose.yml:**

|  |
| --- |
| version: "3.6"    networks:  backend-net:  volumes:  backend-service-logs:    services:    payroll-provider-service:  image: payroll-provider-service  container\_name: payroll-provider-service\_container  build:  context: .  dockerfile: Dockerfile  ports:  - "9000:9000"  environment:  MONGO\_INITDB\_DATABASE: luthersales  MONGO\_INITDB\_ROOT\_USERNAME: user  networks:  - backend-net  volumes:  - /data/backend-service-logs:/log |

**Dev Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/payroll-provider-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9000  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**UAT Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/payroll-provider-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9000  CMD ["java","-Dspring.profiles.active=uat","-jar","app.jar"] |

**Prod Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/payroll-provider-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9000  CMD ["java","-Dspring.profiles.active=prod","-jar","app.jar"] |

**2.1.9: notification-service**

**Dev and UAT and docker-compose.yml:**

|  |
| --- |
| version: "3.6"    networks:  backend-net:  volumes:  backend-service-logs:    services:    notification-service:  image: notification-service  container\_name: notification-service\_container  build:  context: .  dockerfile: Dockerfile  ports:  - "9020:9020"  environment:  MONGO\_INITDB\_DATABASE: luthersales  MONGO\_INITDB\_ROOT\_USERNAME: user  networks:  - backend-net  volumes:  - /data/backend-service-logs:/log |

**Dev Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/notification-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9020  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**UAT Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/notification-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9020  CMD ["java","-Dspring.profiles.active=uat","-jar","app.jar"] |

**Prod Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/notification-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9020  CMD ["java","-Dspring.profiles.active=prod","-jar","app.jar"] |

**2.1.10: helper-service**

**Dev and UAT and docker-compose.yml:**

|  |
| --- |
| version: "3.6"    networks:  backend-net:  volumes:  backend-service-logs:    services:    helper-service:  image: helper-service  container\_name: helper-service\_container  build:  context: .  dockerfile: Dockerfile  ports:  - "9090:9090"  environment:  MONGO\_INITDB\_DATABASE: luthersales  MONGO\_INITDB\_ROOT\_USERNAME: user  networks:  - backend-net  volumes:  - /data/backend-service-logs:/log  - /data/frontend-service-logs:/clientLog |

**Dev Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/helper-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9090  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**UAT Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/helper-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9090  CMD ["java","-Dspring.profiles.active=uat","-jar","app.jar"] |

**Prod Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/helper-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9090  CMD ["java","-Dspring.profiles.active=prod","-jar","app.jar"] |

**2.1.11: new-payroll-provider-service**

**Dev and UAT and docker-compose.yml:**

|  |
| --- |
| version: "3.6"    networks:  backend-net:  volumes:  backend-service-logs:    services:    new-payroll-provider-service:  image: new-payroll-provider-service  container\_name: new-payroll-provider-service\_container  build:  context: .  dockerfile: Dockerfile  ports:  - "9080:9080"  environment:  MONGO\_INITDB\_DATABASE: luthersales  MONGO\_INITDB\_ROOT\_USERNAME: user  networks:  - backend-net  volumes:  - /data/backend-service-logs:/log |

**Dev Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/new-payroll-provider-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9080  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**UAT Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/new-payroll-provider-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9080  CMD ["java","-Dspring.profiles.active=uat","-jar","app.jar"] |

**Prod Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/new-payroll-provider-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9080  CMD ["java","-Dspring.profiles.active=prod","-jar","app.jar"] |

**2.1.12: verification-link-service**

**Dev and UAT and docker-compose.yml:**

|  |
| --- |
| version: "3.6"    networks:  backend-net:  volumes:  backend-service-logs:    services:    verification-link-service:  image: verification-link-service  container\_name: verification-link-service\_container  build:  context: .  dockerfile: Dockerfile  ports:  - "9040:9040"  environment:  MONGO\_INITDB\_DATABASE: luthersales  MONGO\_INITDB\_ROOT\_USERNAME: user  networks:  - backend-net  volumes:  - /data/backend-service-logs:/log |

**Dev Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/link-verification-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9040  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**UAT Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/link-verification-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9040  CMD ["java","-Dspring.profiles.active=uat","-jar","app.jar"] |

**Prod Dockerfile:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/link-verification-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9040  CMD ["java","-Dspring.profiles.active=prod","-jar","app.jar"] |

**2.2 Ingress file**

|  |
| --- |
| apiVersion: extensions/v1beta1  kind: Ingress  metadata:  name: payuw-ingress  annotations:  kubernetes.io/ingress.class: alb  alb.ingress.kubernetes.io/scheme: internet-facing  alb.ingress.kubernetes.io/target-type: ip  alb.ingress.kubernetes.io/scheme: internet-facing  alb.ingress.kubernetes.io/security-groups: sg-09960e517541935ab  spec:  rules:  - host:  http:  paths:  - path: /\*  backend:  serviceName: eureka-server  servicePort: 8761  - path: /notification-service/\*  backend:  serviceName: notification-service  servicePort: 8080  - path: /accounts-management-service/\*  backend:  serviceName: accounts-management-service  servicePort: 8080  - path: /helper-service/\*  backend:  serviceName: helper-service  servicePort: 8080  - path: /new-payroll-provider-service/\*  backend:  serviceName: new-payroll-provider-service  servicePort: 8080  - path: /order-management-service/\*  backend:  serviceName: order-management-service  servicePort: 8080  - path: /otp-service/\*  backend:  serviceName: otp-service  servicePort: 8080  - path: /payroll-provider-service/\*  backend:  serviceName: payroll-provider-service  servicePort: 8080  - path: /transaction-management-service/\*  backend:  serviceName: transaction-management-service  servicePort: 8080  - path: /user-management-service/\*  backend:  serviceName: user-management-service  servicePort: 8080  - path: /verification-link-service/\*  backend:  serviceName: verification-link-service  servicePort: 8080 |

**2.2 Helm Charts for each service:**

**Configmap:**

Create a configmap to externalize the properties present in application.properties and application yaml file.

How to add key values in configmap for spring boot application?

For example, to store DB URL in configmap, instead of simply specifying url, you should specify fully qualified path to key like “multitenancy.mtapp.master.datasource.url” and provide its value separated by a colon. Likewise specify all key-value pairs under data section inside configmap.

**Service account:**

In Kubernetes, service accounts are used to provide an identity for pods. These are used to provide appropriate permissions for pods to communicate with other K8s resources like configmaps, services, endpoints, etc.

**Deployments:**

As we will push images to Amazon container registry (ECR), properly tag the image according to the login server name of your ECR instance.

Note: service.yaml, configmap.yaml and deployment.yaml is same for all the Backend service

Text

Description automatically generated with low confidence

**service.yaml**

|  |
| --- |
| kind: Service  apiVersion: v1  metadata:  name: {{ template "serviceId.name" . }}  labels:  app: {{ template "serviceId.name" . }}  chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"  release: "{{ .Release.Name }}"  namespace: "{{ .Release.Namespace }}"  spec:  type: NodePort  ports:  - port: {{ .Values.service.port }}  targetPort: {{ .Values.service.targetPort }}  protocol: {{ .Values.service.protocol | default "TCP" }}  selector:  app: {{ template "serviceId.name" . }}  release: "{{ .Release.Name }}" |

**configmap.yaml**

|  |
| --- |
| apiVersion: v1  kind: ConfigMap  metadata:  name: {{ template "serviceId.name" . }}  labels:  chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"  release: "{{ .Release.Name }}"  heritage: "{{ .Release.Service }}"  namespace: "{{ .Release.Namespace }}"  data:  {{- with .Values.configMap }}  {{- toYaml . | nindent 2 }}  {{- end -}} |

**deployment.yaml**

|  |
| --- |
| apiVersion: apps/v1  kind: Deployment  metadata:  name: {{ template "serviceId.name" . }}  labels:  app: {{ template "serviceId.name" . }}  chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"  release: "{{ .Release.Name }}"  heritage: "{{ .Release.Service }}"  namespace: "{{ .Release.Namespace }}"  spec:  replicas: 1  selector:  matchLabels:  app: {{ template "serviceId.name" . }}  release: {{ .Release.Name }}  template:  metadata:  labels:  app: {{ template "serviceId.name" . }}  chart: "{{ .Chart.Name }}-{{ .Chart.Version }}"  release: "{{ .Release.Name }}"  heritage: "{{ .Release.Service }}"  namespace: "{{ .Release.Namespace }}"  spec:  containers:  - name: {{ template "serviceId.name" . }}  image: "{{ .Values.imageRepository }}:{{ .Values.imageTag }}"  imagePullPolicy: {{ .Values.imagepullPolicy }}  resources:  {{ toYaml .Values.resources | indent 12 }}  ports:  - name: http  containerPort: 8080  volumeMounts:  - name: {{ template "serviceId.name" . }}-configmap  mountPath: /config/  envFrom:  - configMapRef:  name: {{ template "serviceId.name" . }}  # livenessProbe:  # httpGet:  # path: /{{ template "serviceId.name" . }}/health  # port: http  # initialDelaySeconds: {{ .Values.livenessProbe.initialDelaySeconds }}  # periodSeconds: 20  # timeoutSeconds: 10  # failureThreshold: 10  # readinessProbe:  # httpGet:  # path: /{{ template "serviceId.name" . }}/health  # port: http  # initialDelaySeconds: {{ .Values.readinessProbe.initialDelaySeconds }}  # periodSeconds: 20  # timeoutSeconds: 10  # failureThreshold: 10  volumes:  - name: {{ template "serviceId.name" . }}-configmap  configMap:  name: {{ template "serviceId.name" . }} |

**2.2.1: accounts-management**

**Chart.yaml**

|  |
| --- |
| apiVersion: v2  name: accounts-management  version: 1.0  description: accounts-management |

**dev.yaml (This is value.yaml for dev env)**

|  |
| --- |
| service:  port: 80  targetPort: 8080    imageRepository: "669171879954.dkr.ecr.us-east-1.amazonaws.com/accounts-management-service"  imageTag: "86a9aa6"  imagepullPolicy: Always  resources:  limits:  cpu: 500m  memory: 1Gi  requests:  cpu: 250m  memory: 500M  readinessProbe:  initialDelaySeconds: 120  livenessProbe:  initialDelaySeconds: 120  configMap:  spring.profiles.active: dev  SERVER\_PORT: "8080"  MONGODB\_Host: mongodb://mongo-mongodb-headless:27017/luthersales  LYONS\_API\_BASEURL: https://demo.lyonsreg.com/webservices/aoa/AOAServiceWCF.svc/  LYONS\_API\_COMPANYID: "2785"  LYONS\_API\_USERNAME: inhousecapitaldemo  LYONS\_API\_PASSWORD: "zpCW7Vm7"  LYONS\_API\_RETURNDETAILS: "0"  FINARECT\_API\_BASEURL: https://fineract/fineract-provider/api/v1/  FINARECT\_API\_USERNAME: mifos  FINARECT\_API\_PASSWORD: password  FINARECT\_API\_LOCALE: en  FINARECT\_API\_DATEFORMAT: dd MMMM yyyy  APPLICATION\_NAME: accounts-management  EUREKA\_SERVICEURL\_DEFALTZONE: http://eureka-server:8761/eureka |

**2.2.2: helper**

**Chart.yaml**

|  |
| --- |
| apiVersion: v2  name: helper  version: 1.0  description: helper |

**dev.yaml (This is value.yaml for dev env)**

|  |
| --- |
| service:  port: 80  targetPort: 8080    imageRepository: "669171879954.dkr.ecr.us-east-1.amazonaws.com/helper-service"  imageTag: "1bc56fc"  imagepullPolicy: Always  resources:  limits:  cpu: 500m  memory: 1Gi  requests:  cpu: 250m  memory: 500M  readinessProbe:  initialDelaySeconds: 120  livenessProbe:  initialDelaySeconds: 120  configMap:  spring.profiles.active: dev  SERVER\_PORT: "8080"  APPLICATION\_NAME: helper  EUREKA\_SERVICEURL\_DEFALTZONE: http://eureka-server:8761/eureka |

**2.2.3: new-payroll-provider**

**Chart.yaml**

|  |
| --- |
| apiVersion: v2  name: new-payroll-provider  version: 1.0  description: new-payroll-provider |

**dev.yaml (This is value.yaml for dev env)**

|  |
| --- |
| service:  port: 80  targetPort: 8080    imageRepository: "669171879954.dkr.ecr.us-east-1.amazonaws.com/new-payroll-provider-service"  imageTag: "d70f735"  imagepullPolicy: Always  resources:  limits:  cpu: 500m  memory: 1Gi  requests:  cpu: 250m  memory: 500M  readinessProbe:  initialDelaySeconds: 120  livenessProbe:  initialDelaySeconds: 120  configMap:  spring.profiles.active: dev  SERVER\_PORT: "8080"  MONGODB\_Host: mongodb://mongo-mongodb-headless:27017/luthersales  ARGYLE\_BASE\_LINK: https://api-sandbox.argyle.io/link/v1/  SERVICE\_ID: Salary Allocation  NEW\_PAYROLL\_SERVICE: http://NEW-PAYROLL-PROVIDER-SERVICE/api/v1  ATOMICFI\_ACCOUNT\_NUMBER: 30909999  ATOMICFI\_ROUNTING\_NUMBER: 121000248  ATOMICFI\_ACCOUNTS\_TYPE: checking  ATOMICFI\_ACCOUNTS\_TITLE: Premier Plus Checking  ATOMICFI\_BASE: https://sandbox-api.atomicfi.com/  ATOMICFI\_API\_KEY: 626fb572-b945-4ea1-81a9-b2d8bc7ebac2  ATOMICFI\_SECRET: eb28fd6b-fa28-40f3-aed1-6d52f4916065  ATOMICFI\_TOKEN\_VALIDITY\_HRS: 24  APPLICATION\_NAME: new-payroll-provider  DEFAULT\_URL: http://eureka-server:8761/eureka |

**2.2.4: notification**

**Chart.yaml**

|  |
| --- |
| apiVersion: v2  name: notification  version: 1.0  description: notification |

**dev.yaml (This is value.yaml for dev env)**

|  |
| --- |
| service:  port: 80  targetPort: 8080  imageRepository: "669171879954.dkr.ecr.us-east-1.amazonaws.com/notification-service"  imageTag: "1.1"  imagepullPolicy: Always  resources:  limits:  cpu: 500m  memory: 1Gi  requests:  cpu: 250m  memory: 500M  readinessProbe:  initialDelaySeconds: 120  livenessProbe:  initialDelaySeconds: 120  configMap:  spring.profiles.active: dev  SERVER\_PORT: "8080"  MONGODB\_Host: mongodb://mongo-mongodb-headless:27017/luthersales  TWILLIO.ACCOUNTSID: ACe5cd554ad621b5e5571fe90f71b45c7f  TWILLIO.AUTHTOKEN: d8fe1e6633f44cc2bd0acc2bbff61af1  APPLICATION\_NAME: notification  DEFAULT\_URL: http://eureka-server:8761/eureka |

**2.2.5: order-management**

**Chart.yaml**

|  |
| --- |
| apiVersion: v2  name: order-management  version: 1.0  description: order-management |

**dev.yaml (This is value.yaml for dev env)**

|  |
| --- |
| service:  port: 80  targetPort: 8080  imageRepository: "669171879954.dkr.ecr.us-east-1.amazonaws.com/order-management-service"  imageTag: "9e31935"  imagepullPolicy: Always  resources:  limits:  cpu: 500m  memory: 1Gi  requests:  cpu: 250m  memory: 500M  readinessProbe:  initialDelaySeconds: 120  livenessProbe:  initialDelaySeconds: 120  configMap:  spring.profiles.active: dev  SERVER\_PORT: "8080"  MONGODB\_Host: mongodb://mongo-mongodb-headless:27017/luthersales  BASELINK\_URL: https://sandbox1-dev.paywalletllc.com/order/  PAYROLL\_EUREKA\_URL: http://PAYROLL-PROVIDER-SERVICE/api/v1/  NOTIFICATION\_EUREKA\_URL: http://NOTIFICATION-SERVICE/api/v1/}  VERIFICATION\_EUREKA\_URL: http://LINK-SERVICE/api/v1/}  NEWPAYROLL\_EUREKA\_URL: http://NEW-PAYROLL-PROVIDER-SERVICE/api/v1/}  EMAIL\_BODY: Purchase Order Email Body  CUSTOMER\_TEMPLATEiD: d-35c08b1fc0b24b34b3394df4ab0d1fa9  CUSTOMER\_LINK\_SUBJECT: Your LutherSales purchase  REP\_LINK\_SUBJECT: Luthersales purchase initiation for  REP\_LINK\_TEMPLATEID: d-e6c99da5466041378a91d99cdf99d459  CUSTOMER\_SUCCESS\_TEMPLATEID: d-ed1c053bb8594c13950b2d7878ce40b2  CUSTOMER\_SUCCESS\_SUBJECT: Your LutherSales Purchase status  REP\_SUCCESS\_SUBJECT: Luthersales transaction status for  REP\_SUCCESS\_TEAMPLATEID: d-6421923ae20742cf9c33200299da41b6  REP\_DECLINE\_TEMPLATEID: d-a76b1140b0784523a225e2bfeb8b0f5d  CUSTOMER\_DECLINE\_SUBJECT: Your LutherSales Purchase status  REP\_DECLINE\_SUBJECT: Luthersales transaction status for  REP\_DECLINE\_TEMPLATEID: d-659ab3c303094553a31c948d8bfb8e75  SMS\_LINK\_TEMPLATE: To complete your purchase from Luther Sales using installment deduct of $%s directly from salary for %s pay periods, please click on the link below within %s hours and authorize direct payment of installment amount to Luther Sales. Thank you - Luther Sales Team \n%s  SMS\_ACCEPT\_TEMPLATE: Your installment deduction of $%s per pay period will occur for %s payperiods towards payments for the purchased item cost. Thank you - Luther Sales team  SMS\_REJECT\_TEMPLATE: You declined payment using installment payments from your payroll for the purchase of $%s. Please contact us at 800-358-6466 for assistance to complete the purchase using alternate payment methods - LutherSales Team  SMS\_PENDING\_ALLOCATION: Your pay allocation request is pending approval from your employer %s  SMS\_FAILD\_ALLOCATION: Not Enough money left to complete your message  CUSTOMER\_ALLOCATION\_FAILED\_TEMPLATEID: d-0ca34e45493b4ddc80fa69e29c84dd51  CUSTOMER\_ALLOCATION\_PENDING\_TEMPLATEID: d-dd8a6e01f97846f38a0079105f8fcfb1  CUSTOMER\_ALLOCATION\_SUBJECT: Status of allocation  REP\_ALLOCATION\_FAILED\_TEMPLATEID: d-de3a0be28c4d490d88c9ace24943fe4e  REP\_ALLOCATION\_PENDING\_TEMPLATEID: d-3fdcd0c0bc134470952cecea314d9f76  PAGE\_SIZE: "10"  PAGE\_NO: "0"  ABA\_NUMBER: "121000248"  APPLICATION\_NAME: order-management  DEFAULT\_URL: http://eureka-server:8761/eureka |

**2.2.6: otp**

**Chart.yaml**

|  |
| --- |
| apiVersion: v2  name: otp  version: 1.0  description: otp |

**dev.yaml (This is value.yaml for dev env)**

|  |
| --- |
| service:  port: 80  targetPort: 8080  imageRepository: "669171879954.dkr.ecr.us-east-1.amazonaws.com/otp-service"  imageTag: "211faf5"  imagepullPolicy: Always  resources:  limits:  cpu: 500m  memory: 1Gi  requests:  cpu: 250m  memory: 500M  readinessProbe:  initialDelaySeconds: 120  livenessProbe:  initialDelaySeconds: 120  configMap:  spring.profiles.active: dev  SERVER\_PORT: "8080"  MONGODB\_Host: mongodb://mongo-mongodb-headless:27017/luthersales  OTP\_EXPIRY\_TIME: 10  NOTIFICATION\_EUREKA\_URL: http://NOTIFICATION-SERVICE  NOTIFICATION\_SMS\_URL: /api/v1/notifications/sms  NOTIFICATION\_EMAIL\_SERVICE: /api/v1/notifications/email  NOTIFICATION\_SMS\_TEMPLATE: LutherSales: Otp is the single use security code you requested. This will expire in ${otp.expiryTimeInMins} minutes. Do not share it- We will never contact you to ask for it  KEY\_CLOAK: luther-sales  KEY\_CLOAK\_AUTH\_SERVER\_URL: https://sandbox1-dev.paywalletllc.com/auth/realms/  KEY\_CLOAK\_CLIENTID: kong}  KEY\_CLOAK\_CLENT\_SECRET: db2a2703-2e33-4186-9652-20db7d9024e8  TWILLIO.ACCOUNTSID: ACe5cd554ad621b5e5571fe90f71b45c7f  TWILLIO.AUTHTOKEN: d8fe1e6633f44cc2bd0acc2bbff61af1  APPLICATION\_NAME: otp  DEFAULT\_URL: http://eureka-server:8761/eureka |

**2.2.7: payroll-provider**

**Chart.yaml**

|  |
| --- |
| apiVersion: v2  name: payroll-provider  version: 1.0  description: payroll-provider |

**dev.yaml (This is value.yaml for dev env)**

|  |
| --- |
| service:  port: 80  targetPort: 8080    imageRepository: "669171879954.dkr.ecr.us-east-1.amazonaws.com/payroll-provider-service"  imageTag: "b7ad7d6"  imagepullPolicy: Always  resources:  limits:  cpu: 500m  memory: 1Gi  requests:  cpu: 250m  memory: 500M  readinessProbe:  initialDelaySeconds: 120  livenessProbe:  initialDelaySeconds: 120  configMap:  spring.profiles.active: dev  SERVER\_PORT: "8080"  MONGODB\_Host: mongodb://mongo-mongodb-headless:27017/luthersales  CONTEXT\_PATH: /api/v1/\*  MAPPING\_SERVLET: CamelServlet  EMPLOYERS\_CONNECTION: mongodb:mongoClient?database=luthersales&collection=ArgyleEmployers&hosts=mongo-mongodb-headless:27017  ARGYLE\_TOKEN: e2a46c5e5145492392fd0b59a7d0731f:d6dadcb80e296efa04fc10d5a3b70ba7d9e5153c5ac97218c7cf040b25afbffd  ARGYLE\_BASE: https://api-sandbox.argyle.io/v1/  ARGYLE\_LINK: https://api-sandbox.argyle.io/link/v1/  ARGYLE\_PLUGINKEY: 114b8dfa-dd26-4ae9-b2e0-cd1c838a2503  ARGYLE\_FEATURES: eyJwYXlfZGlzdHJpYnV0aW9uIjp0cnVlLCJtZmFfY29uZmlybWF0aW9uX3VybF9zZW50X2J5X2VtYWlsIjp0cnVlLCJwZHNfbWZhIjp0cnVlfQ==  AGGREGATOR\_PREFERRED: ARGYLE  AGGREGATOR\_SERVICE: Pay-Wallet  FINERACT\_EXTERNALID\_PREFIX: DPOC2-  FINERACT\_CUSTOMER\_ROUTINGNO: "123456789"  FINERACT\_CUSTOMER\_PAYMENTTYPEID: "2"  JWT\_SECRET: eyJwYXlfZGlzdHJpYnV0aW9uIjp0cnVlLCJtZmFfY29uZmlybWF0aW9uX3VybF9zZW50X2J5X2VtYWlsIjp0cnVlLCJwZHNfbWZhIjp0cnVlfQ==  WELLSFARGO\_CUSTOMER: "121000248"  APPLICATION\_NAME: payroll-provider  EUREKA\_SERVICEURL\_DEFAULTZONE: http://eureka-server:8761/eureka |

**2.2.8: transaction-management**

**Chart.yaml**

|  |
| --- |
| apiVersion: v2  name: transaction-management  version: 1.0  description: transaction-management |

**dev.yaml (This is value.yaml for dev env)**

|  |
| --- |
| service:  port: 80  targetPort: 8080    imageRepository: "669171879954.dkr.ecr.us-east-1.amazonaws.com/transaction-management-service"  imageTag: "9ed4862"  imagepullPolicy: Always  resources:  limits:  cpu: 500m  memory: 1Gi  requests:  cpu: 250m  memory: 500M  readinessProbe:  initialDelaySeconds: 120  livenessProbe:  initialDelaySeconds: 120  configMap:  spring.profiles.active: dev  SERVER\_PORT: "8080"  MONGODB\_Host: mongodb://mongo-mongodb-headless:27017/luthersales  FINERACT\_BASE\_URL: http://fineract/fineract-provider/api/v1/  INERACT\_USER\_NAME: mifos  FINERACT\_PASSWORD: password  FINERACT\_LOCALE: en  FINERACT\_DATE\_FORMAT: dd MMMM yyyy  APPLICATION\_NAME: TransactionService  DEFAULT\_URL: http://eureka-server:8761/eureka |

**2.2.9: user-management**

**Chart.yaml**

|  |
| --- |
| apiVersion: v2  name: user-management  version: 1.0  description: user-management |

**dev.yaml (This is value.yaml for dev env)**

|  |
| --- |
| service:  port: 80  targetPort: 8080    imageRepository: "669171879954.dkr.ecr.us-east-1.amazonaws.com/user-management-service"  imageTag: "f7f46f9"  imagepullPolicy: Always  resources:  limits:  cpu: 500m  memory: 1Gi  requests:  cpu: 250m  memory: 500M  readinessProbe:  initialDelaySeconds: 120  livenessProbe:  initialDelaySeconds: 120  configMap:  spring.profiles.active: dev  SERVER\_PORT: "8080"  MONGODB\_Host: mongodb://mongo-mongodb-headless:27017/luthersales  LYONS\_BASE\_URL: https://demo.lyonsreg.com/webservices/aoa/AOAServiceWCF.svc/  LYONS\_COMPANY\_ID: 2785  LYONS\_USER\_NAME: inhousecapitaldemo  LYONS\_PASSWORD: zpCW7Vm7  LYONS\_RETURN\_DETAILS: 0  CUST\_VALIDATION\_MAX\_ALLOWED\_UPDATE: 3  APPLICATION\_NAME: user-management  DEFAULT\_ZONE: http://eureka-server:8761/eureka |

**2.2.10: verification-link**

**Chart.yaml**

|  |
| --- |
| apiVersion: v2  name: verification-link  version: 1.0  description: verification-link |

**dev.yaml (This is value.yaml for dev env)**

|  |
| --- |
| service:  port: 80  targetPort: 8080    imageRepository: "669171879954.dkr.ecr.us-east-1.amazonaws.com/verification-link-service"  imageTag: "55cf0e8"  imagepullPolicy: Always  resources:  limits:  cpu: 500m  memory: 1Gi  requests:  cpu: 250m  memory: 500M  readinessProbe:  initialDelaySeconds: 120  livenessProbe:  initialDelaySeconds: 120  configMap:  spring.profiles.active: dev  SERVER\_PORT: "8080"  MONGODB\_Host: mongodb://mongo-mongodb-headless:27017/luthersales  LINK\_EXPIRYTIME: "4320"  OTP\_EUREKA\_URI: http://OTP-SERVICE  APPLICATION\_NAME: verification-link  EUREKA\_CLIENT\_SERVICEURL: http://eureka-server:8761/eureka |

**Jenkins**

Jenkins Job Configuration

* Here we are using <https://jenkins.paywalletllc.com/> server
* First, configure AWS container registry (ECR) credentials
* Slack notification is configured. On paywallet\_builds channel all the jenkins job status is notified

Create a Jenkins job of type “pipeline” and in the pipeline section copy the below pipeline accordingly.

1. **Build-Backend-Image-Pipeline**

Choose “This job is parameterized” and create one string parameters for “**Branch**” , two being “**Service**” and “**environment/namespace”**.

**For Dev:  
Parameters:**

Branch - Enter the branch to Build

Service - Select the service you want to Build

Environment - Select the environment you want to deploy

|  |
| --- |
| pipeline {  agent any  environment {  BUILD\_TRIGGER\_BY = "${currentBuild.getBuildCauses()[0].shortDescription}"  AWS\_ACCOUNT\_ID="669171879954"  AWS\_DEFAULT\_REGION="us-east-1"  GIT\_COMMIT\_REV=''  REPOSITORY\_URI = "${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com"  }  parameters {  string(name: 'Branch', defaultValue: 'temp-migration', description: 'Enter the branch to deploy?')  choice(name: 'Service', choices: ['new-payroll-provider-service', 'payroll-provider-service', 'accounts-management-service', 'user-management-service', 'order-management-service', 'helper-service', 'transaction-management-service', 'verification-link-service', 'otp-service', 'notification-service', 'eureka-service'], description: 'Select the service you want to deploy')  choice(name: 'environment', choices: ['dev'], description: 'Select the environment you want to deploy')  }  stages {  stage('Logging into AWS ECR') {  steps {  script {  sh "aws ecr get-login-password --region ${AWS\_DEFAULT\_REGION} | docker login --username AWS --password-stdin ${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com"  }  }  }  stage('Cloning our Git') {  steps {  git branch: '$Branch', credentialsId: 'Github-Key', url: 'https://github.com/Maveric-Digital/${Service}.git'  }  }    stage('Configure') {  steps {  script {  GIT\_COMMIT\_REV = sh(returnStdout: true, script: "git log -n 1 --pretty=format:'%h'").trim()  }  }  }    stage ('Maven Build') {  steps {  sh """  mvn clean install -DskipTests  """  }  }    // Building Docker images  stage('Building image') {  steps{  script {  dockerImage = docker.build "${Service}:${GIT\_COMMIT\_REV}"  }  }  }  // Uploading Docker images into AWS ECR  stage('Pushing to ECR') {  steps{  script {  sh "docker tag ${Service}:$GIT\_COMMIT\_REV ${REPOSITORY\_URI}/${Service}:$GIT\_COMMIT\_REV"  sh "docker push ${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com/${Service}:$GIT\_COMMIT\_REV"  sh "docker rmi ${Service}:$GIT\_COMMIT\_REV ${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com/${Service}:$GIT\_COMMIT\_REV"  }  }  }  }  post {  success {  slackSend (color: '#00FF00', message: "Successfully pushed the ${Service} dokcer image for ${environment} environment, ${BUILD\_TRIGGER\_BY}. Docker image version for this build is $GIT\_COMMIT\_REV' (${env.BUILD\_URL})")  }  failure {  slackSend (color: '#FF0000', message: "Failed to pushed the ${Service} dokcer image for ${environment} environment, ${BUILD\_TRIGGER\_BY}. Docker image version for this build is $GIT\_COMMIT\_REV' (${env.BUILD\_URL})")  }  always {  cleanWs()  }  }  } |

1. **Build-Frontend-Image-Pipeline**

Choose “This job is parameterized” and create one string parameters for “**Branch**” , two being “**Service**” and “**environment/namespace”**.

**For Dev:  
Parameters:**

Branch - Enter the branch to Build

Service - Select the service you want to Build

Environment - Select the environment you want to deploy

|  |
| --- |
| pipeline {  agent any  environment {  BUILD\_TRIGGER\_BY = "${currentBuild.getBuildCauses()[0].shortDescription}"  AWS\_ACCOUNT\_ID="669171879954"  AWS\_DEFAULT\_REGION="us-east-1"  GIT\_COMMIT\_REV=''  REPOSITORY\_URI = "${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com"  }  parameters {  string(name: 'Branch', defaultValue: 'migration', description: 'Enter the branch to deploy?')  choice(name: 'Service', choices: ['luther-sales-salesrep-app', 'luther-sales-borrower-app'], description: 'Select the service you want to deploy')  choice(name: 'environment', choices: ['dev'], description: 'Select the environment you want to deploy')  }  stages {  stage('Logging into AWS ECR') {  steps {  script {  sh "aws ecr get-login-password --region ${AWS\_DEFAULT\_REGION} | docker login --username AWS --password-stdin ${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com"  }  }  }  stage('Cloning our Git') {  steps {  git branch: '$Branch', credentialsId: 'Github-Key', url: 'https://github.com/Maveric-Digital/${Service}.git'  }  }    stage('Configure') {  steps {  script {  GIT\_COMMIT\_REV = sh(returnStdout: true, script: "git log -n 1 --pretty=format:'%h'").trim()  }  }  }    // Building Docker images  stage('Building image') {  steps{  script {  dockerImage = docker.build "${Service}:${GIT\_COMMIT\_REV}"  }  }  }  // Uploading Docker images into AWS ECR  stage('Pushing to ECR') {  steps{  script {  sh "docker tag ${Service}:$GIT\_COMMIT\_REV ${REPOSITORY\_URI}/${Service}:$GIT\_COMMIT\_REV"  sh "docker push ${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com/${Service}:$GIT\_COMMIT\_REV"  sh "docker rmi ${Service}:$GIT\_COMMIT\_REV ${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com/${Service}:$GIT\_COMMIT\_REV"  }  }  }  }  post {  success {  slackSend (color: '#00FF00', message: "Successfully pushed the ${Service} dokcer image for ${environment} environment, ${BUILD\_TRIGGER\_BY}. Docker image version for this build is $GIT\_COMMIT\_REV' (${env.BUILD\_URL})")  }  failure {  slackSend (color: '#FF0000', message: "Failed to pushed the ${Service} dokcer image for ${environment} environment, ${BUILD\_TRIGGER\_BY}. Docker image version for this build is $GIT\_COMMIT\_REV' (${env.BUILD\_URL})")  }  always {  cleanWs()  }  }  } |

1. **paywallet-deployer**

Choose “This job is parameterized” and create one string parameters for “**Docker\_tag**” , two being “**Service**” and “**environment/namespace”**.

**For Dev:  
Parameters:**

Service - Select the service you want to deploy

Environment - Select the environment you want to deploy

Docker\_tag - Enter the docker image version you want to deploy(On slack channel docker image build is published)

|  |
| --- |
| pipeline {  agent any  environment {  BUILD\_TRIGGER\_BY = "${currentBuild.getBuildCauses()[0].shortDescription}"  }  parameters {  choice(name: 'Service', choices: ['accounts-management', 'eureka', 'helper', 'new-payroll-provider', 'notification', 'order-management', 'otp', 'payroll-provider', 'transaction-management', 'user-management', 'verification-link'], description: 'Select the service you want to deploy')  choice(name: 'environment', choices: ['dev'], description: 'Select the environment you want to deploy')  string(name: 'Docker\_tag', description: 'Enter the Docker image version to deploy')  }  stages {  stage('Cloning our helm chart for the service') {  steps {  git credentialsId: 'Github-Key', url: 'https://github.com/Maveric-Digital/devops-config.git'  }  }  stage('Deploy the helm chart for the service ') {  steps{  script {  sh "helm upgrade --install ${Service} helm/${Service} --values helm/${Service}/${environment}.yaml --set imageTag=${Docker\_tag} -n ${environment}"  }  }  }  }  post {  success {  slackSend (color: '#00FF00', message: "Deployed ${Service}-service in ${environment} environment ${BUILD\_TRIGGER\_BY} (${env.BUILD\_URL})")  }  failure {  slackSend (color: '#FF0000', message: "Deployment failed for ${Service}-service in ${environment} environment ${BUILD\_TRIGGER\_BY} (${env.BUILD\_URL})")  }  always {  cleanWs()  }  }  } |

**Paywallet**

**GCP environment configuration Guide**

**GCP-Architecture**

Chart

Description automatically generated

**Dev-Environment Architecture**

Diagram

Description automatically generated

**UAT-Environment Architecture**

Graphical user interface, diagram, application

Description automatically generated

**Prod-Environment**

Diagram

Description automatically generated

List of micro-services that are running in GCP enviornment.

* luther-sales-borrower-app
* luther-sales-salesrep-app
* new-payroll-provider-service
* order-management-service
* notification-service
* payroll-provider-service
* accounts-management-service
* user-management-service
* helper-service
* transaction-management-service
* verification-link-service
* luthersales-inhouse-app-php-code
* otp-service
* eureka-service
* mongo-db

Note: All the above services and docker files are updated in the **2.1 Containerization** Section.

Some of the other services Dockerfile and docker-compose are updated in the below.

**Dockerfile and docker-compose for each service:**

For Backend Service: To run the micro-services in containers with Docker, first clone the repository from Github and build it (mvn clean install -DskipTests). Now write a Dockerfile to copy the resultant war file into container and run it using java –jar command. Place the Dockerfile under root directory of your project.

For Frontend Service: As the frontend is ReactJS application. Build the npm package on the on-prem server and copy the resultant build to docker image.

**kong-keycloak:**

**Dev and UAT and docker-compose.yml:**

|  |
| --- |
| version: '3.4'  networks:  kong-net:  keycloak-net:  volumes:  kong-datastore:  keycloak-datastore:  keycloak-tls:  services:  kong-db:  image: postgres:11-alpine  volumes:  - kong-datastore:/var/lib/postgresql/data  networks:  - kong-net  ports:  - "15432:5432"  environment:  POSTGRES\_DB: api-gw  POSTGRES\_USER: kong  POSTGRES\_PASSWORD: kong  kong:  build:  context: ./  image: kong-oidc:latest  depends\_on:  - kong-db  networks:  - kong-net  ports:  - "80:80" # Listener  - "8001:8001" # Admin API  - "443:8443" # Listener (SSL)  - "8444:8444" # Admin API (SSL)  environment:  KONG\_DATABASE: postgres  KONG\_PG\_HOST: kong-db  KONG\_PG\_PORT: 5432  KONG\_PG\_DATABASE: api-gw  KONG\_PG\_USER: kong  KONG\_PG\_PASSWORD: kong  KONG\_PROXY\_ACCESS\_LOG: /dev/stdout  KONG\_ADMIN\_ACCESS\_LOG: /dev/stdout  KONG\_PROXY\_ERROR\_LOG: /dev/stderr  KONG\_ADMIN\_ERROR\_LOG: /dev/stderr  KONG\_PROXY\_LISTEN: 0.0.0.0:80, 0.0.0.0:8443 ssl  KONG\_ADMIN\_LISTEN: 0.0.0.0:8001, 0.0.0.0:8444 ssl  KONG\_PLUGINS: bundled,oidc  KONG\_LOG\_LEVEL: debug  KONG\_NGINX\_PROXY\_PROXY\_BUFFER\_SIZE: 512k  KONG\_NGINX\_PROXY\_PROXY\_BUFFERS: 64 512k  konga:  image: pantsel/konga:0.14.7  depends\_on:  - kong  networks:  - kong-net  ports:  - "1337:1337" # konga  environment:  DB\_ADAPTER: postgres  DB\_HOST: kong-db  DB\_PORT: '5432'  DB\_USER: kong  DB\_PASSWORD: kong  DB\_DATABASE: api-gw  NODE\_ENV: development  keycloak-db:  image: postgres:11-alpine  volumes:  - keycloak-datastore:/var/lib/postgresql/data  networks:  - keycloak-net  ports:  - "25432:5432"  environment:  POSTGRES\_DB: keycloak  POSTGRES\_USER: keycloak  POSTGRES\_PASSWORD: password  keycloak:  image: jboss/keycloak:14.0.0  depends\_on:  - keycloak-db  networks:  - keycloak-net  ports:  - "8180:8080"  - "8443:8443"  environment:  DB\_VENDOR: POSTGRES  DB\_ADDR: keycloak-db  DB\_PORT: 5432  DB\_DATABASE: keycloak  DB\_USER: keycloak  DB\_PASSWORD: password  KEYCLOAK\_USER: admin  KEYCLOAK\_PASSWORD: admin  PROXY\_ADDRESS\_FORWARDING: 'true'  volumes:  - keycloak-tls:/etc/x509/https |

**Prod docker-compose.yml:**

|  |
| --- |
| version: '3.4'  networks:  kong-net:  keycloak-net:  volumes:  kong-datastore:  keycloak-datastore:  keycloak-tls:  services:  kong:  build:  context: ./  image: kong-oidc:latest  networks:  - kong-net  ports:  - "80:80" # Listener  - "8001:8001" # Admin API  - "443:8443" # Listener (SSL)  - "8444:8444" # Admin API (SSL)  restart: always  environment:  KONG\_DATABASE: postgres  KONG\_PG\_HOST: 10.4.0.2 #kong-db  KONG\_PG\_PORT: 15432  KONG\_PG\_DATABASE: api-gw  KONG\_PG\_USER: kong  KONG\_PG\_PASSWORD: kong  KONG\_PROXY\_ACCESS\_LOG: /dev/stdout  KONG\_ADMIN\_ACCESS\_LOG: /dev/stdout  KONG\_PROXY\_ERROR\_LOG: /dev/stderr  KONG\_ADMIN\_ERROR\_LOG: /dev/stderr  KONG\_PROXY\_LISTEN: 0.0.0.0:80, 0.0.0.0:8443 ssl  KONG\_ADMIN\_LISTEN: 0.0.0.0:8001, 0.0.0.0:8444 ssl  KONG\_PLUGINS: bundled,oidc  KONG\_LOG\_LEVEL: debug  KONG\_NGINX\_PROXY\_PROXY\_BUFFER\_SIZE: 512k  KONG\_NGINX\_PROXY\_PROXY\_BUFFERS: 64 512k  konga:  image: pantsel/konga:0.14.7  depends\_on:  - kong  networks:  - kong-net  ports:  - "1337:1337" # konga  restart: always  environment:  DB\_ADAPTER: postgres  DB\_HOST: 10.4.0.2 # kong-db  DB\_PORT: 15432 # '5432'  DB\_USER: kong  DB\_PASSWORD: kong  DB\_DATABASE: api-gw  NODE\_ENV: development  keycloak:  image: jboss/keycloak:14.0.0  depends\_on:  - konga  networks:  - keycloak-net  ports:  - "8180:8080"  - "8443:8443"  restart: always  environment:  DB\_VENDOR: POSTGRES  DB\_ADDR: 10.4.0.2 # keycloak-db  DB\_PORT: 25432 # 5432  DB\_DATABASE: keycloak  DB\_USER: keycloak  DB\_PASSWORD: password  KEYCLOAK\_USER: admin  KEYCLOAK\_PASSWORD: Paywallet@123  PROXY\_ADDRESS\_FORWARDING: 'true'  volumes:  - keycloak-tls:/etc/x509/https |

**Database:**

**Prod env mongo-db docker-compose.yml:**

|  |
| --- |
| version: '3.5'  services:  mongodb:  container\_name: mongo-db  image: mongo:latest  environment:  - MONGO\_INITDB\_ROOT\_USERNAME=admin  - MONGO\_INITDB\_ROOT\_PASSWORD=password@123  - MONGO\_INITDB\_DATABASE=luthersales  ports:  - '27017-27019:27017-27019'  volumes:  - mongo:/data/db  - ./init-mongo.js:/docker-entrypoint-initdb.d/init-mongo.js:ro  - /data/mongodb:/data  networks:  - mongodb  restart: unless-stopped  networks:  mongodb:  driver: bridge    volumes:  mongo: |

**Prod env postgresql-db docker-compose.yml:**

|  |
| --- |
| version: '3.5'  services:  keycloak-db:  image: postgres:11-alpine  volumes:  - keycloak-datastore:/var/lib/postgresql/data  networks:  - keycloak-net  ports:  - "25432:5432"  environment:  POSTGRES\_DB: keycloak  POSTGRES\_USER: keycloak  POSTGRES\_PASSWORD: password  kong-db:  image: postgres:11-alpine  volumes:  - kong-datastore:/var/lib/postgresql/data  networks:  - kong-net  ports:  - "15432:5432"  environment:  POSTGRES\_DB: api-gw  POSTGRES\_USER: kong  POSTGRES\_PASSWORD: kong  networks:  kong-net:  driver: bridge  keycloak-net:  driver: bridge    volumes:  kong-datastore:  keycloak-datastore: |

**Jenkins Configuration:**

**Dev – Env:**

**Dev:** <https://jenkins-dev.paywalletllc.com/>

**Username:** Paywallet

**Password:** Paywallet@123

**UAT:** <https://jenkins-uat.paywalletllc.com/>

**Username:** Paywallet

**Password:** Paywalletuat@123

**Prod:** <https://jenkins-prod.paywalletllc.com/>

**Username:** Paywallet

**Password:** Paywalletprod@123

Pipeline for Dev-env services: <https://github.com/Maveric-Digital/devops-config/tree/master/Dev-env>

Pipeline for UAT-env services: <https://github.com/Maveric-Digital/devops-config/tree/master/uat-env>

Pipeline for Prod-env services: <https://github.com/Maveric-Digital/devops-config/tree/master/prod-env>