FICO® Decision Management Platform

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

In

Computer Science and Engineering Department

By

Shikhar Goel (151357)

Under the supervision of

Dr Hemraj Saini

To

Department of Computer Science & Engineering

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

FICO®

Fair Isaac India Software Pvt. Ltd
CIN U72200KA2003PTC032938
Registered Office:
Titanium Building
135 Airport Road
Kodihalli, Bangalore 560 017, India
T 91 80 4137 1700
F 91 80 4137 1701
Email: investor@fico.com
www.fico.com

INTERNSHIP CERTIFICATE

This is to certify that **Shikhar Goel** s/o **Adv Parveen Goel**, a student of **B.TECH** Roll No. 151357, "Jaypee University Of Information Technology, Solan" is undergoing his internship program at our company from 2ndFeburary 2019 to 30thJune 2019.

During his internship, Shikhar has closely worked as a part of **Decision Management Platform (DMP)Developer** team in FICO. In addition, has been working under the guidance of his supervisor **Mr Shaik Mujeeb** to successfully complete this project. The performance during this period of his internship program with us was good. He was punctual, hardworking and inquisitive to learn new technologies.

Shaik Mujeeb

Software Engineering-Lead Engineer,

DMS Development

Fair Isaac India Software Pvt. Ltd

Date-

DECLARATION

I hereby declare that the work which is being presented in this project report entitled "**FICO® Decision Management Platform**", in partial fulfillment of the requirement for the award of the degree of **Bachelor of technology** submitted to Department of Computer Science & Engineering and Information Technology (CSE&IT) in JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, Solan is an authentic work done by me during a period from 2nd Feburary, 2019 to 30th June, 2019 under the Guidance of **Trupti Vaishnav, Shaik Mujeeb, Santosh Kumar, Siddartha, Ritu Rani and Surbhi Jain**in **FICO.**

The work presented in this project report has not been submitted by me for the award of any other degree of this or any other Institute/University.

Signature

Shikhar Goel

151357

This is to certify that the above statement made by the candidate is correct to best of my knowledge and belief.

Date : **Signature**

Place : **Shaik Mujeeb**

Software Engineering-Lead Engineer

ABSTRACT

In months of internship at Fair Isaac Corporation (FICO), I was hired as a Software Engineering Intern for the Products and Technology Organization (PTO) team. I was assigned to Software Engineering profile under Decision Management Platform. I was responsible for writing code for setting up cluster; executing the code and managing all activities in the code base to ensure that all the objectives are met and that the solution works as expected. The software should be tested in terms of functionality, performance, reliability, stability, and compatibility with another legacy- and/or external systems. We was required to dockerize and deploy various DMP functionality. We wote many ansible scripts and helm charts to automate the process of cluster set up.We deployed these functionality using software like Java, Kubernetes, Openshift, Jenkins.

Keywords:Deployment, Jenkins, Openshift, Kubernetes

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

Chapter 1
INTRODUCTION

This first chapter is mainly to introduce the reader to the FICO, as well as the basic structure of this project report. In this chapter, the reader can get introduced to the organization, FICO, under the heading 1.1. The reader would know about the major clients of the organization from the 1.2. Under 1.3. Heading reader would get the structure of this thesis.

1.1.COMPANY PROFILE

FICO (NYSE: FICO) is main investigation programming organization, helping organizations in 90+ nations settle on better choices that drive more elevated amounts of development, gainfulness, and consumer loyalty. The organization's historical utilization of Big Data and numerical calculations to foresee purchaser conduct has changed whole commercial enterprises.

FICO gives examination programming and devices utilized over numerous commercial ventures to oversee hazard, battle misrepresentation, manufacture more productive client connections, advance operations and meet strict government regulations. A number of our items achieve extensive appropriation —, for example, the FICO® Score, the standard measure of purchaser credit hazard in the United States. FICO arrangements influence open-source guidelines and distributed computing to expand adaptability, speed organization,and decrease costs. The organization likewise offers a great many individuals some assistance with managing their own credit wellbeing. FICO: Make each choice count™.

Established in 1956, FICO presented expository arrangements, for example, credit scoring that has made credit all the more generally accessible in the United States as well as around the globe. We have spearheaded the advancement and utilization of basic innovations behind the choice administration. These incorporate prescient examination, business rules administration and improvement. We utilize these advancements to offer organizations some assistance with improving the accuracy, consistency, and spryness of their complex, high–volume choices.

1.2. FICO CLIENTS: FICO clients include more than half of the top 100 banks in the world, more than 600 personal and commercial line insurers in North America and Europe including the top

Figure 1. 1 Fico Customers

10 US personal lines insurers, 400+ retailers and general merchandisers, including one-third of the top 100 U.S. retailers, 95 of the 100 largest financial institutions in the U.S., and all the 100 largest U.S. credit card.Issuers and more.

Chapter 2

PROJECT UNDERTAKEN

2.1 OBJECTIVE

This project is aimed at providing a set of cloud-based services for rapidly creating, deploying, and operationalizing a wide range of decision and analytic services. This project explainsa softwareapplication that uses decision services such as business rules, predictive models, and optimization models to solve business problems. It handlessecurity, scalability, lifecycle management, and data integration for your solutions and their components.

2.2 PROJECT MOTIVATION

- This decision is done by the company according to their needs but main aim of project is to learn new technology and required functionalities with passion and get our hands on the work required for the company.
- To know about theexisting software and Software Development lifecycle.
- To know how to design and develop software systems, using scientific analysis and mathematical models to predict and measure outcome and consequences of design.
- To know how to analyze information to determine, recommend and plan installation of a new system or modification of an existing system.
- To understand how to design new test cases.
- To learn about the various technology used like Kubernetes, docker etc.
- To learn about how to perform Unit testing and Functional testing.

2.3 INDUSTRY APPLICATION

Our organization Fair Isaac Corporation (FICO), will get help from this project in various ways:

- Deliver best products to the customer in less time.
- Reduce the burden of the manager.
- Get a new talent.
- Reduced development time of the project.
- To work on different projects.

- Contribution towards revenue
- Attract more customers
- Help banks and other organization to make right decisions

Chapter 3

LITERATURE REVIEW: FICO DECISION MANAGEMENT SUITE

3.1 INTRODUCTION

The FICO® Decision Management Suite gives a simple approach to clients to assess, customize, deploy and scale cutting edge investigation and choice administration arrangements. It permits clients to rapidly coordinate FICO and FICO accomplice choice administration instruments and segments with their own particular segments, offering associations of all sizes some assistance with realizing the guarantee of cutting edge examination and choice administration by means of savvy, adaptable cloud and on-premises arrangements.

FICO decision management suite follows the most recent technological four layered architecture to solve problems. These layers are –Analytics, Rules, Optimization, and Rapid Application & Workflow Development tool. Big data model is processed through all four layers and it helps us in decision-making.

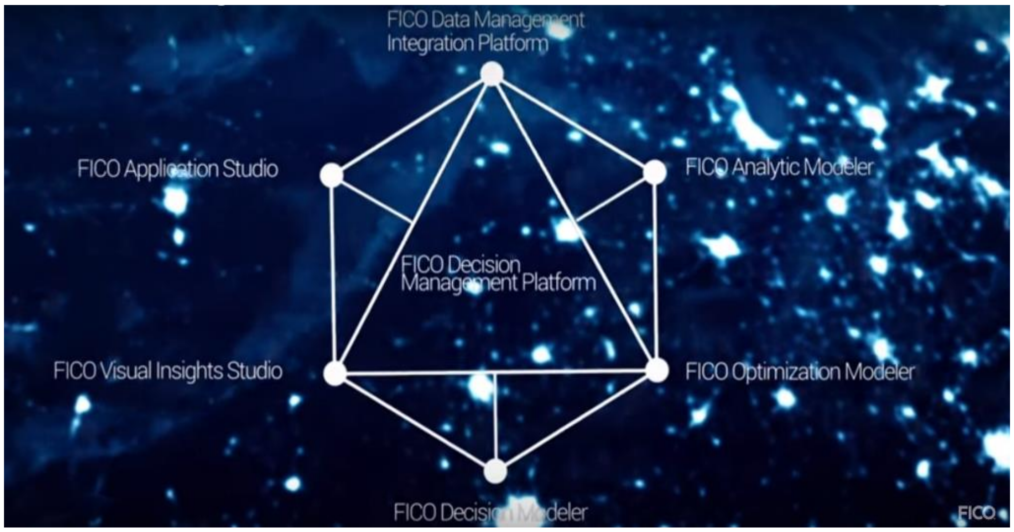

FICO Data Management Integration Platform
FICO Application Studio
FICO Analytic Modeler
FICO Decision Management Platform
FICO Visual Insights Studio
FICO Optimization Modeler
FICO Decision Modeler
FICO

Figure 3.1 Big Data Model

To improve customer satisfaction and faster deeper relationships with FICO Decision Management Suite, one can make real-time choices that impact one’s customers immediately. Decision Management Tool allow to collect data from multiple sources and create analytic based model to predict customer behavior. It helps one to engage with their customers with more reliability and can create more opportunities. All those customers' data one can actually prepare his future strategies of his business and products. The various customers of FICO are- Healthcare, Insurance, Banking, Retail, Banking and others. In healthcare, it reduces the cost with the highest possible care for patients. It saves insurance companies from fraud, errors, abuse and also helps to reduce the loss with customize conditions for each policy holder. It helps banking sectors to identify correct decision through adopting the economic changes over the real time analysis of market strategies. It helps the retail customers to identify their customer needs and also helps in advanced marking of merchandises. It helps Govt. sectors to improve the national security system or to secure the public interests. Decision management solution is an automated, refine and connect high volume business decisions which can be identified by analytical calculation or predictive analysis, or optimization.

3.1.1 Architecture of FICO Decision Management Suite:

The FICO Decision Management Suite, alongside the FICO Analytic Cloud, gives an exhaustive domain that makes it brisk and simple to gather bits of knowledge from information, and create systematic models and choice administrations that operationalize those bits of knowledge. It flawlessly coordinates examination, decisioning, improvement, information perception and investigation, quick application advancement and other FICO Decision Management Platform capacities to give a complete and dexterous deaccessioning arrangement.

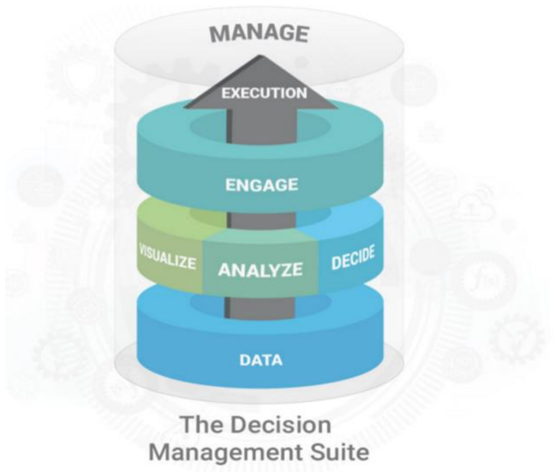


Figure 3.2Decision management suite

3.2 FICO ANALITCS CLOUD

The adaptability and availability of the cloud permits new business chances to be caught quicker than any time in recent memory. Prescient investigation and bleeding edge applications that used to take up to a year to create can now be in clients' grasp inside of months. That is the reason it is added to the FICO® Analytic Cloud, to help rapidly unravel the most complex business challenges without the weight of legacy limitations. FICO has taken demonstrated choice administration arrangements and capable information science instruments and made them accessible in a protected cloud environment so client can make and send cutting edge examination speedier than at any other time.

Crosswise over numerous associations big data and data science are driving a quickly extending pipeline of new application advancement ventures. Progressively, prescient examination are at the focal point of these new, inventive use cases. All things considered, it's these examination that give the capacity to decipher and suspect business needs. At the point when infused into business work processes, progressed examination can significantly enhance the exactness of strategic and vital choices.

That is the reason big data is a major ordeal. Not as a result of the volume of data being made, but since of the prescient force that can now be refined and quickly connected. The FICO Analytic Cloud organizes data science, business rationale and improvement methods in the cloud so individuals can rapidly move examination out of the lab and into the applications that serve bleeding edge choice making.

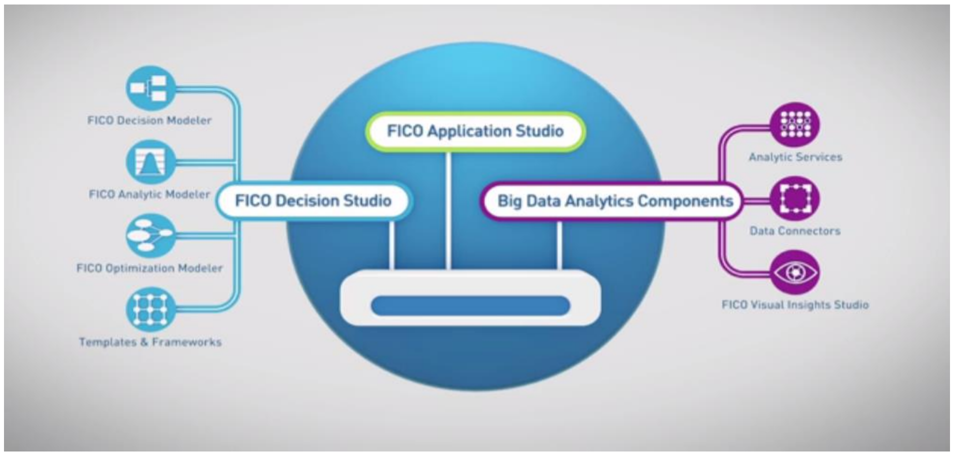


Figure 3.3 FICO Application Studio

3.2.1 Architecture of FICO Analytic Cloud:

It is an end-to-end infrastructure and ecosystem that includes on-demand hardware, tools, SaaS applications, vertical solutions, community and market place for Analytics and Decision Management.

It has four major parts and they work together to provide value:

- **FICO Platform/Solution Stack**

  It helps FICO, its partners and its customers to build solution rapidly, build new solution, solve new business problem.
- **FICO Solutions**

  There are some products that are already built, helps companies to solve their business problems.
- **Market Place**

  An open place for customers, developers, partners to add, sell or customize components or solutions and extension to existing FICO components.
- **Community**

  Here people can interact with each other to collaborate or to understand the products and components they have taken from FICO.

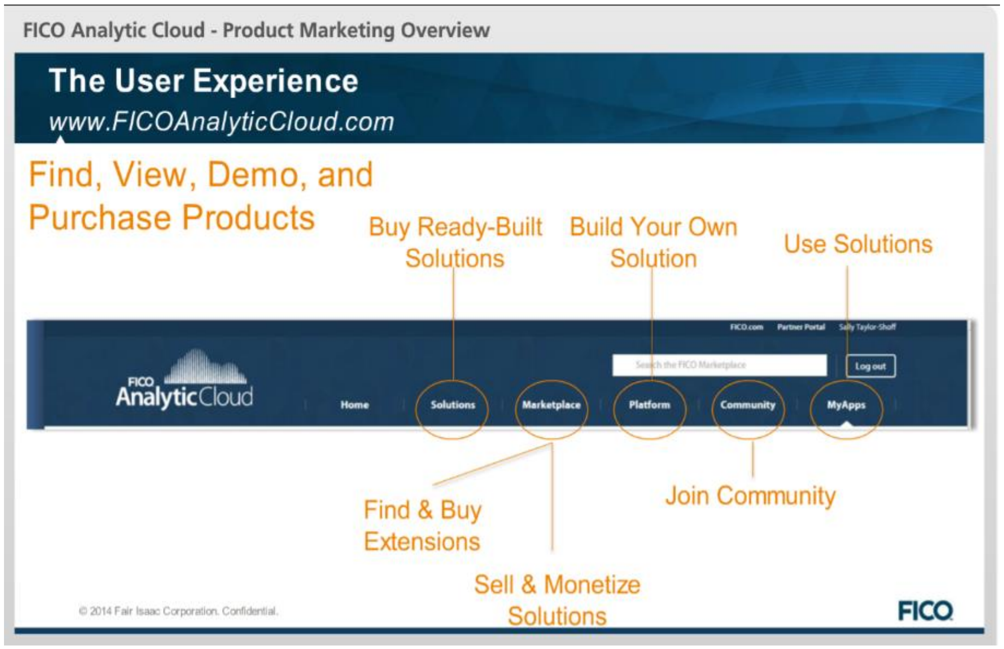

FICO Analytic Cloud - Product Marketing Overview
The User Experience
www.FICOAnalyticCloud.com
Find, View, Demo, and Purchase Products
Buy Ready-Built Solutions
Build Your Own Solution
Use Solutions
FICO AnalyticCloud
Home
Solutions
Marketplace
Platform
Community
MyApps
Log out
Find & Buy Extensions
Sell & Monetize Solutions
Join Community
© 2014 Fair Isaac Corporation. Confidential.
FICO

Figure 3.4Product Marketing Overview

3.3 DECISION MANAGEMENT PLATFORM (DMP)

The Decision Management Platform (DMP) provides secure, managed access to an extensible set of components, services, and execution environments for automating business decisions. DMP is designed to support a sequence of activities that can collectively be referred to as decision management.

Applications for developing models, and business systems that call the decision services, communicate with DMP through APIs. DMP provides lifecycle and administrative services, and also acts as an intermediary layer between the applications and lower-level infrastructure resources provided by the cloud.

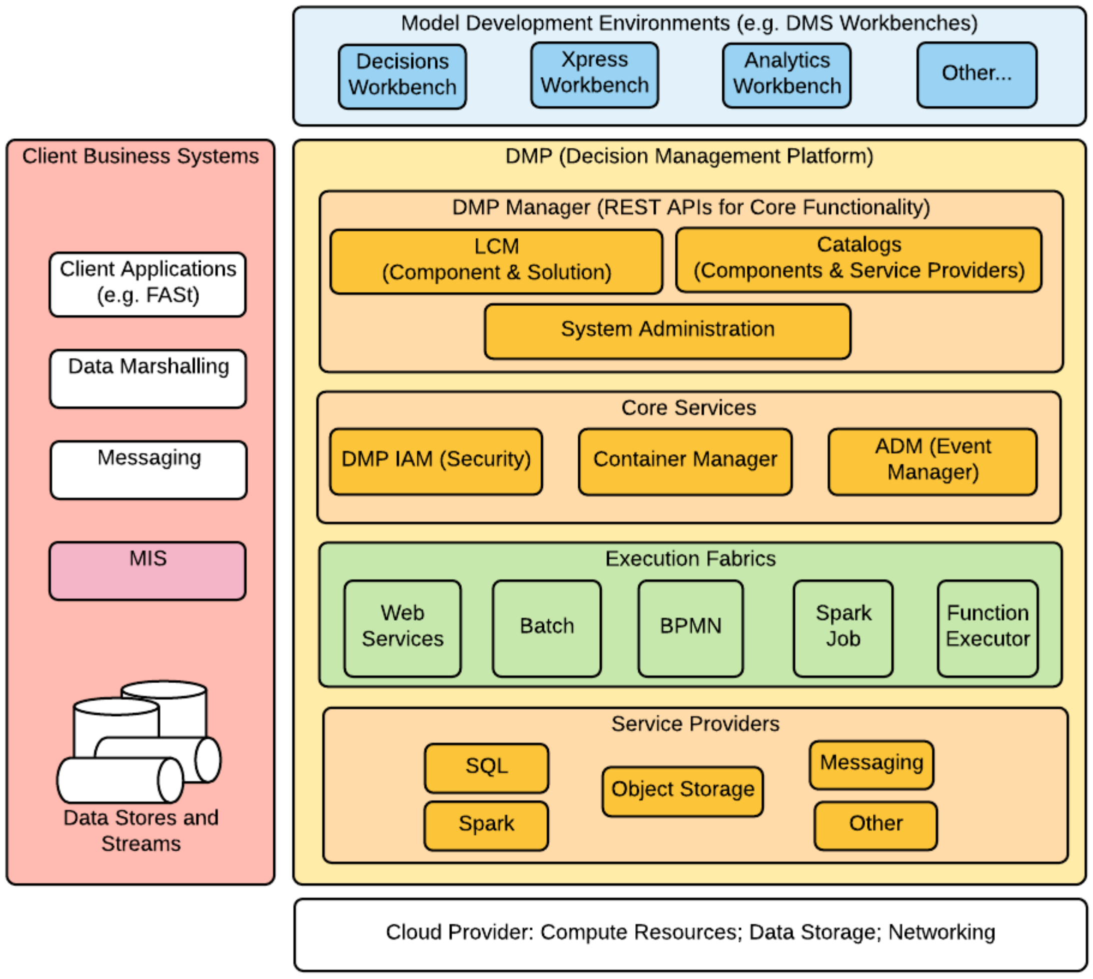


Figure 3.5 System Landscape

DMP exposes APIs that allow external applications to create and manage *solutions*, which are logical containers for *components*, which are packages of intellectual property (IP) that can be deployed in software containers. Components can have dependencies on low-level infrastructure services such as data stores. The infrastructure services and how to connect to them can be discovered dynamically through *service providers*. A DMP installation maintains a catalog of service providers that are configured for that installation. DMP also supports a variety of *execution fabrics* that support packaging components and solutions as executable logic and deploying them to appropriate runtime environments. This section provides definitions and descriptions of solutions, components, and service providers, along with an introduction to the primary DMP subsystems.

3.3.1 Platform Features

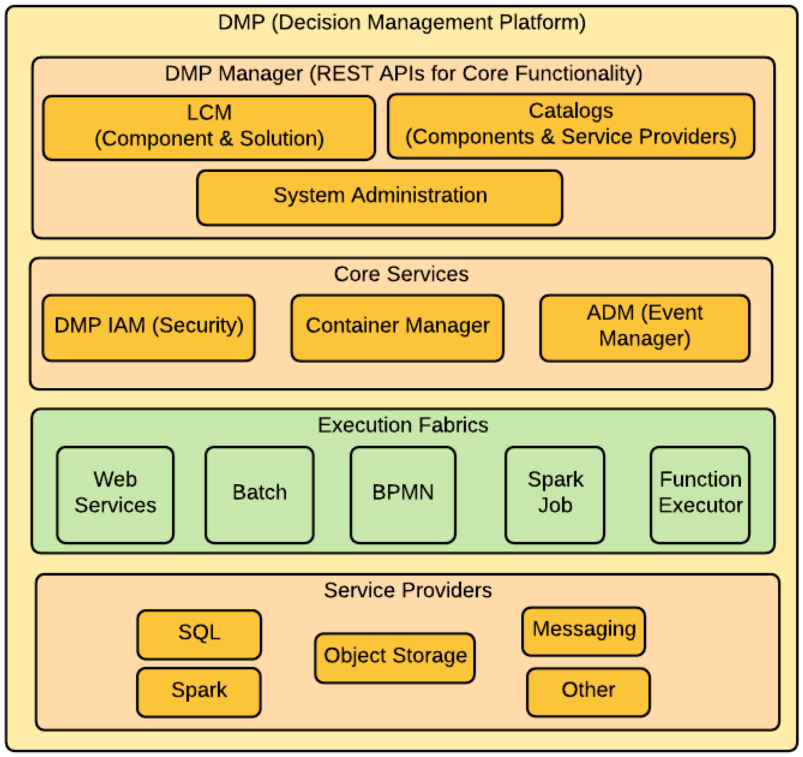


Figure 3.6 Platform Features

3.3.2 DMP Components

The platform must be able to support an extensive catalog of tools that are used by the end-users and applications that rely on DMP. The catalog must be easily extensible, and the tools in the catalog must have easy access to the lower-level services provided by the platform. Modeling applications must be able to request tools from the catalog and be granted access to provisioned instances of the tools. To address these requirements, DMP defines a Component Specification for the interfaces and packaging standards that allow a tool to be added to the DMP catalog and provisioned through the Container Manager. Tools that implement the interfaces and are packaged according to the specification can be added to the DMP component catalog, and are referred to as DMP components.

A component is packaged as a *component archetype*, which can be used to provision one or more *instances* of the component.

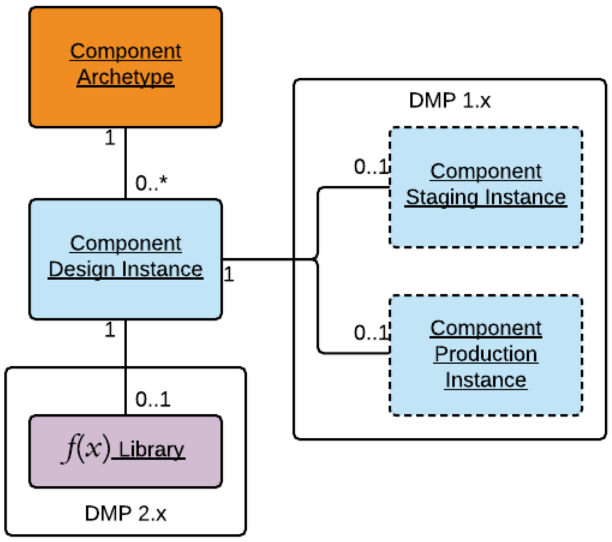

Component Archetype
1
0..*
Component Design Instance
1
1
0..1
f(x) Library
DMP 2.x
DMP 1.x
0..1
Component Staging Instance
0..1
Component Production Instance

Figure 3.7 Component Archetype and Instances

3.3.3 Component Archetypes

A component archetype contains:

- The libraries or other intellectual property required to implement the component.
- Any initial or default configuration.

- A component descriptor that describes what the component is, what it provides in terms of user interfaces and business services, what sort of container it needs to run in (typically a web server), and what services (such as databases, message queues, etc.) it depends on.

The component archetype is uploaded to the DMP, and added to the catalog. DMP Manager APIs enable applications to browse the catalog and request that component instances be provisioned from the archetype.

Component archetypes are separated into two pieces:

- A Docker image, which is stored in a Docker repository
- A package containing a descriptor file (component.json) and a public folder containing other resources such as images and license files.
  These files are stored on object storage so DMP Manager can reference them as raw objects.

3.3.4 Component Instances

When DMP Manager receives a request for the component, it:

- Provisions an appropriate container to run the component instance (the container management service is discussed later).
- Installs the libraries and initial configuration in the container.
- Starts the container and publishes the URLs to the requester.

The first instance of a component is considered the design instance. Any configuration that is supported by the component is done in the design instance. The component descriptor specifies the configuration interfaces that are available.

Components that support deploying configured instances as component web services can be published to staging and production instances. When DMP Manager receives a publishing request, it provisions a new container, with the component IP, but using the configuration from design instance rather than the initial configuration provided with the archetype.

Note: While publishing multiple component instances for executing decision services is still supported, it is being replaced by the more flexible notion of functional deployments.

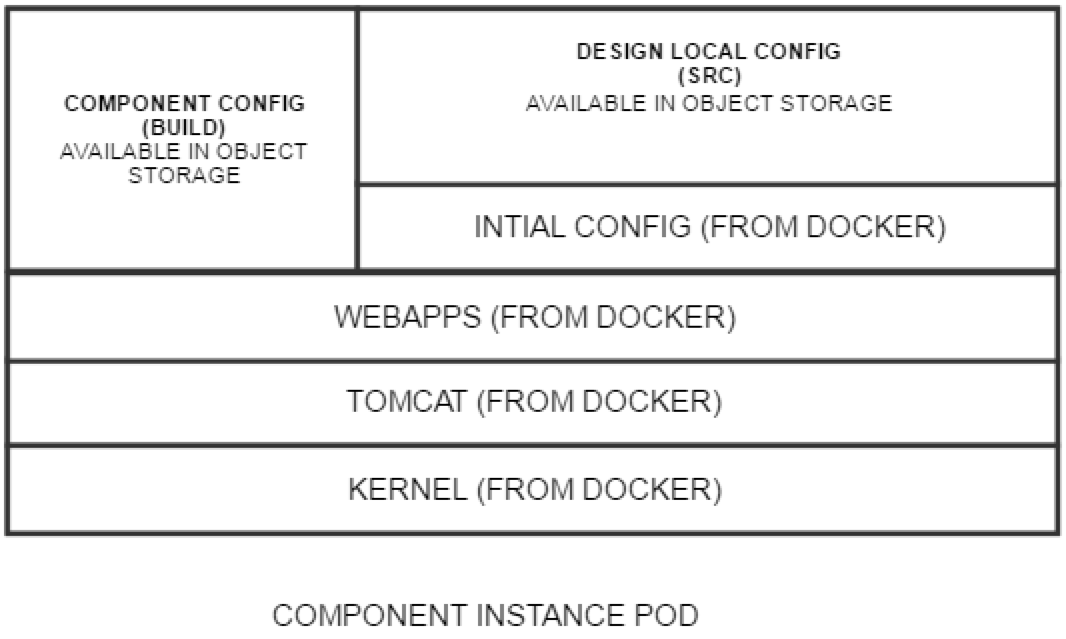


Figure 3.1 Component Instance Container Structure

3.3.5 DMP Functions

DMP functions (referred to in this document simply as “functions”) are libraries of executable code, along with any supporting artifacts required to execute the code. Functions are currently packaged as Java JAR files, although alternative executables may be supported in the future. Functions are generated by DMP component. Functions were introduced in DMP 2.0 to provide a way to support a larger, more extensible, and more flexible set ofExecution Fabrics. Execution fabrics based on Functional Deployments orchestrate functions from one or more function factories into executable services that can be run in a variety of scenarios.

The only way to execute a service on DMP was to invoke over HTTP a web service endpoint that was exposed by a component instance. While component web services continue to be supported, and are suitable for some deployment scenarios, functional deployments support the ability to run a decision service wherever it is needed.

The functions generated by all of the function factories declared by the components in a solution can be managed as a versioned entity called a solution snapshot. Typically, a specific solution snapshot is referenced by any given deployment. How solution snapshots are created and managed is described in Solution Lifecycle.

3.3.6 Service Providers

DMP components can declare dependencies on infrastructure services such as databases, message queues, map/reduce engines, etc. When a component instance is created, it must have a way to dynamically discover and connect to the required services. Service providers are applications that conform to the DMP Service Provider Specification, and which enable components to establish connections to the infrastructure service. DMP Manager maintains a catalog of service providers. When a component is instantiated, DMP Manager checks the catalog for a service provider matching each of the component's declared service dependencies. When an appropriate service provider is found, it is invoked with the component ID. The service provider interacts with the infrastructure service to perform any required actions such as creating a database schema or object storage folder, and then returns applicable connection information. DMP Manager then passes the connection information to the component instance, so it can directly connect to the infrastructure service and interact with it, with no further dependency on DMP Manager.

3.3.7 Solutions

A DMP solution is a named entity with a unique identifier. It is used as a logical container for a set of users, Components and solution services that are managed as a unit. Operations on solutions are performed through the DMP Manager Solution Management APIs

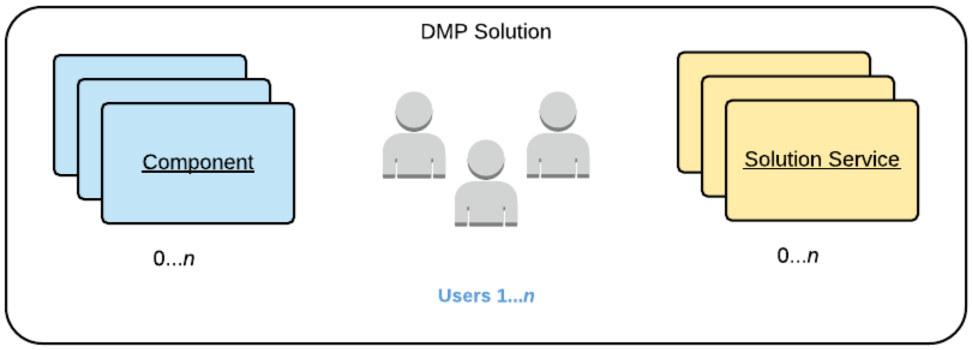


Figure 3.8 Solution structure

Authorized DMP users create and manage solutions, including instantiating and configuring the components associated with the solution. Typically, the user logs into an application such as Solution Builder or Decision Modeler, which manages the calls to the DMP Manager APIs.

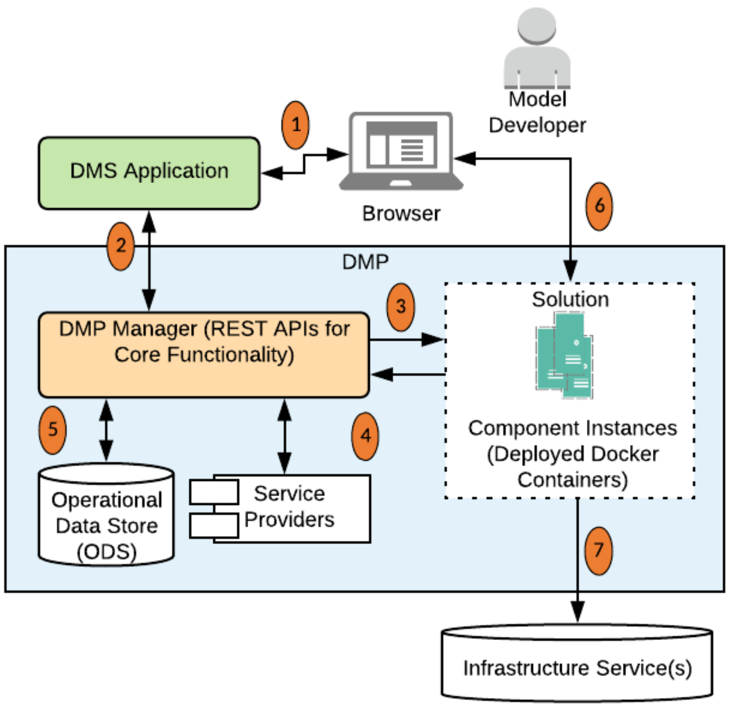

Model Developer
1
Browser
DMS Application
6
2
DMP
3
Solution
DMP Manager (REST APIs for Core Functionality)
Component Instances (Deployed Docker Containers)
5
4
Operational Data Store (ODS)
Service Providers
7
Infrastructure Service(s)

Figure 3.9 Solution Creation Flow

1. The model developer logs into the application through a browser, and manipulates the application interface.
2. The application initiates calls to the DMP Manager APIs for creating and managing solutions, adding components to a solution, and accessing any configuration interfaces exposed by the component.
3. DMP Manager coordinates the completion of the requested operation with the container manager and/or other services.
4. If a component has a dependency on an infrastructure service, DMP Manager queries the service providers catalog and passes the URL to the appropriate provider to the component instance.
5. The operation and relevant data are recorded in the operational data store (ODS)
6. Component configuration interfaces are provided back to the application, so the model developer can configure the component directly.
7. The component instance requests connection information from the provider, and uses it to establish any needed connections to the infrastructure service. Once the connection is established the component instance can interact directly with the service, with no further dependency on DMP Manager.

Solution Components:

A component belongs to exactly one solution. A solution can have any number of components.

Solution Services:

A solution service is an entity that species connection information that can be used by components belonging to a solution to interact with a lower-level infrastructure service. The information is generated by a service provider.

3.3.8 DMP Manager

DMP Manager implements a set of core platform services. These services expose DMP functionality to external applications through REST APIs and implement the interfaces with lower-level subsystems.

TABLE 3.2 **DMP Manager Services**

DMP Manager Service	**Description**
Component Instance Management	Handles requests for creating and managing runnable component instances. Manages versioned snapshots of a component's configuration.
Component Catalog Management	Handles requests for creating and managing component archetypes.
Solutions Management	Associates platform services with the components that comprise a solution. Manages versioned snapshots of functions and artifacts provided by a solution's components.
Service Provider Catalog Management	Maintains and manages the available DMP Services.
Team Management	Provides role-based user authorization for managing components and solutions.
System Administration	Provides access to system configuration and monitoring features.

3.3.9 Component Lifecycle

DMP Manager exposes APIs for creating and managing components. A request for a new component includes the ID of the component archetype.

DMP provides default functionality for migrating a component to multiple LCM environments. APIs are also available to allow custom applications to implement their own lifecycle management. By default, DMP supports the following LCM states:

- **Development** – The initial instance is considered the "development environment," and is used for component configuration and unit testing via its configuration and execution endpoints. The DMP UI provides a menu that exposes configuration interfaces declared in the component descriptor via an embedded iframe; e.g., the Rule Maintenance Application for Blaze-based components.

Any configuration done in the development environment is deployed to any execution environment; e.g., testing and production. For example, rules authored in the development environment of a Blaze-based component are included in the deployments to the execution environments.

- **Staging** – After the configuration process is completed, a user may choose to submit the component to staging. This submission step snapshots the configuration of the component and deploys an instance with the staging configuration. The snapshot is achieved by issuing a REST command from DMP Manager that imports the development instance's data directory into the repository.

  At this point, an independent auditor can approve or reject a component before it goes into production.

- **Production** – After testing has been completed, a component can be promoted to a production state. A snapshot is made of the staging configuration and a new instance is created (or updated) in the Deployment environment.

State transitions are managed by a DMP workflow to provide oversight and traceability. The default workflow can be programmatically customized via public REST APIs.

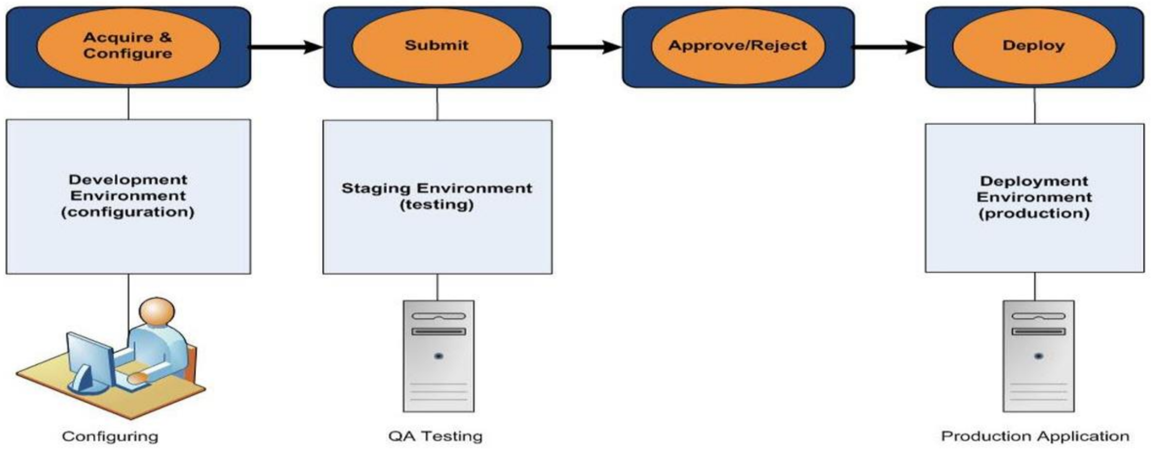

Acquire & Configure
Submit
Approve/Reject
Deploy
Development Environment (configuration)
Staging Environment (testing)
Deployment Environment (production)
Configuring
QA Testing
Production Application

Figure 3.10 Component LCM States

3.3.10 Solution Lifecycle

DMP Manager provides APIs for creating a solution, associating components with the solution, and managing the solution through its lifecycle.

1. **Solution Creation**

When a new solution is requested, DMP Manager creates a new entry in ODS, which includes a unique identifier. The identifier is then used for subsequent operations on the solution.

DMP Manager also makes requests to the S3 service provider to create folders in the S3 bucket for the tenant that will be used to store solution data and ADM event data.

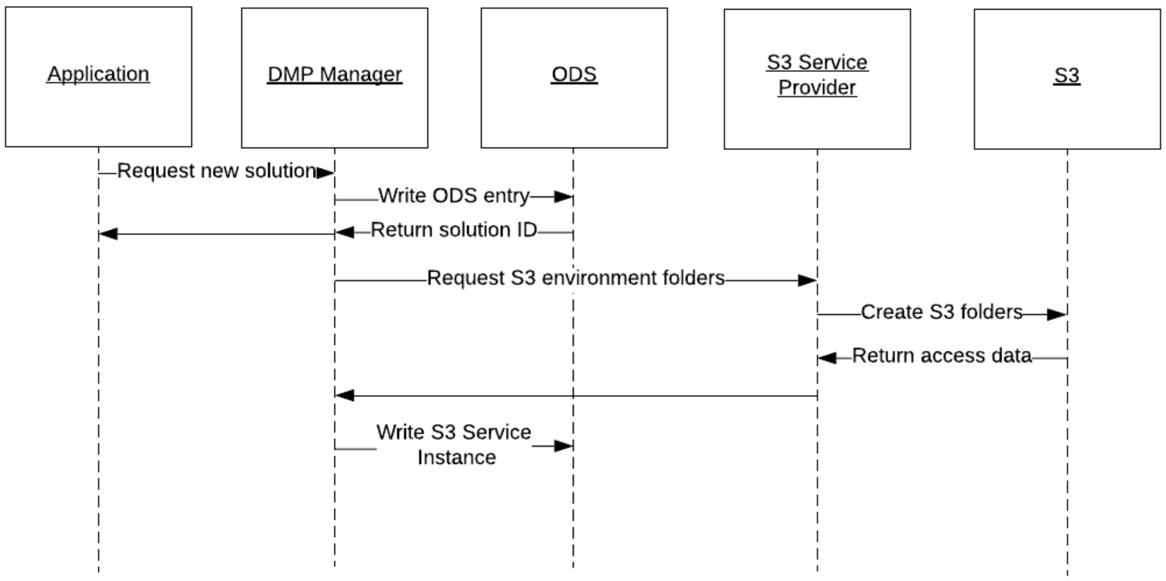


Figure 3.11 Solution Creation

2. **Adding Components**

There are two patterns for adding a component to a solution:

- Select the component type from the component archetype catalog.
- Select a function factory from the factories catalog.

In this case the end-user or application knows the type of function, but may not know about the component that implements the function factory. DMP Manager is responsible for identifying the component that exposes the factory.

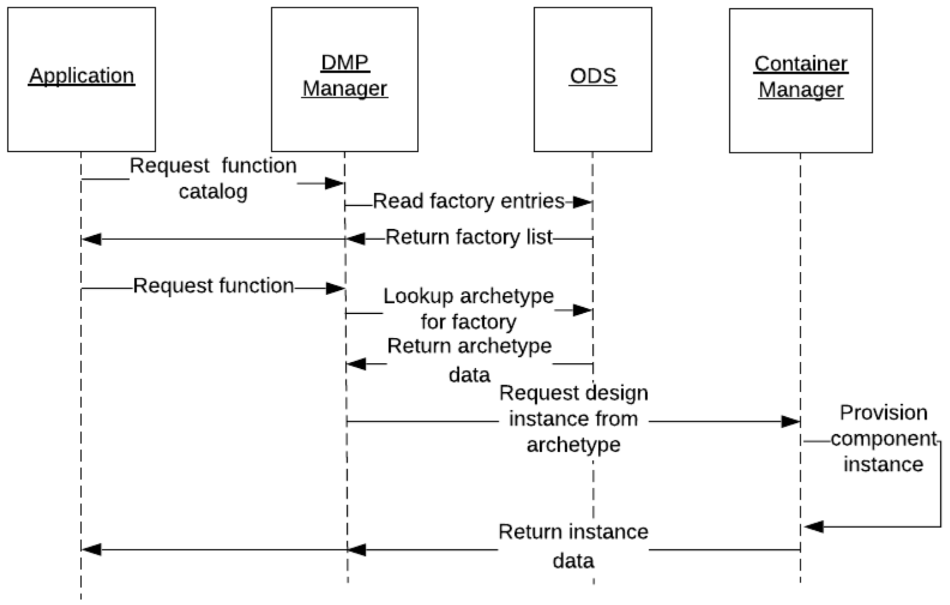


Figure 3.12 Add Component to Solution

3. Solution Commit

A solution has its own lifecycle. It can be *published* as a *solution snapshot*, which is a versioned collection of all of the functions contributed by all of the components that belong to the solution, plus any supporting dependencies or artifacts. Once published, a solution snapshot is immutable until the snapshot is republished. Publishing a solution does not entail provisioning additional component instances. Since the underlying components export their functions as jars, the published solution snapshot consists of a packaged and versioned collection of those jars.

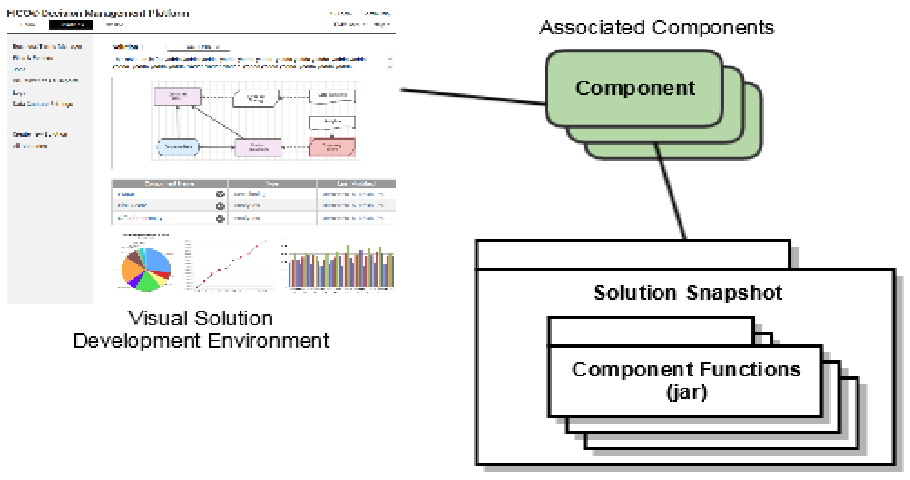


Figure 3.13 Solution-Centric Approach

Committing a solution entails getting all of the function artifacts exposed by all of the components that belong to the solution, and packaging them to a dedicated folder.

Chapter 4

PROPOSED SOLUTION: DMP Laptop Deployment

To give us the understanding of the DMP environment and the technology used by DMP for implementing and maintaining the DMP platform, I have assigned the task to deploy the DMP environment locally on my laptop and then run all the DMP related function such as Solution Creation, building archetypes, Component creation and then performs component lifecycle. This give us the overview of the following technologies:

4.1 DOCKER

Docker could be a tool that's designed to profit each developers and system directors, creating it a vicinity of the many DevOps (developers + operations) toolchains. laborer could be a tool designed to create it easier to make, deploy, and run applications by mistreatment containers. Containers enable a developer to package up Associate in Nursing application with all of the elements it wants, like libraries and alternative dependencies, and ship it all out joined package. By doing therefore, due to the instrumentality, the developer will rest assured that the appliance can run on the other Linux machine despite any bespoke settings that machine might need that would take issue from the machine used for writing and testing the code.

In a way, laborer could be a bit sort of a virtual machine. however not like a virtual machine, instead of making an entire virtual software package, laborer permits applications to use a similar Linux kernel because the system that they are running on and solely needs applications be shipped with things not already running on the host pc. this provides a big performance boost and reduces the scale of the appliance.

4.2 KUBERNETES

DMP laptop deployment can be done in two ways: one by using the openshift environment and other by using Kubernetes (plain vanilla Kubernetes). Vanilla Kubernetes is the most basic installation with all the features of a Kubernetes release. For DMP 3.1 release we are using minikube.

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

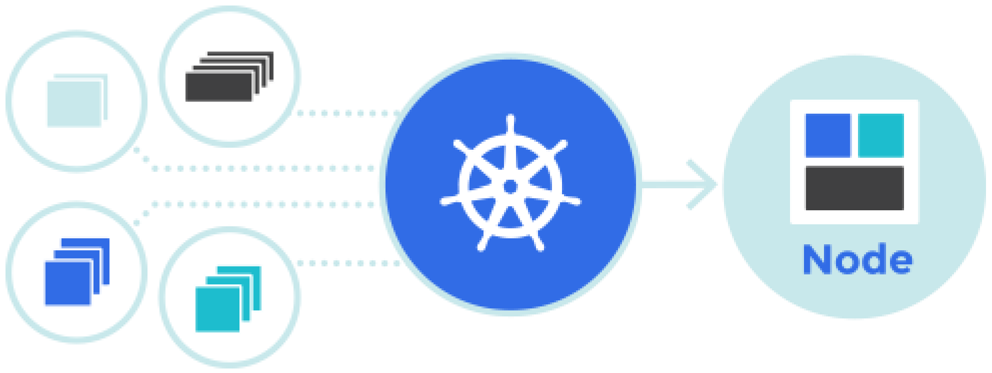


Figure 4.1 Kubernetes

It groups containers that make up an application into logical units for easy management and discovery.

DMP laptop deployment requires following Kubernetes features:

1. **Pod**: (Collection of Containers) A pod is a deployment unit in the K8S with a single IP address. Inside it, the Pause container handles networking by holding a network's namespace, port and ip address, which in turn is used by all containers within the pod.
2. **ReplicationController**: A replication controller ensures that the desired number of containers are up and running at any given time. Pod templates are used to define the container image identifiers, ports, and labels. Using liveness probes, it auto-heals pods and maintains the number of pods as per desired state. It can also be manually controlled by manipulating the replica count using kubectl.
3. **Storage Management**: Pods are ephemeral in nature — any information stored in a pod or container will be lost. In order to store data even after a pod is killed or rescheduled, a persistent system like Amazon Elastic Block Storage (EBS), Google Compute Engine's Persistent Disks (GCE PD), or a distributed file system such as Network File System (NFS) or Gluster File System (GFS) is needed.
4. **Services**: Kubernetes services are abstractions which route traffic to a set of pods to provide a microservice. Kube-proxy runs on each node and manages services by setting up a bunch of iptable rules. There are three modes of setting up services:

a. ClusterIP (only provides access internally)
b. NodePort (needed to open firewall on a port; not recommended for public access)
c. LoadBalancer (owned by public cloud providers like AWS or GKE

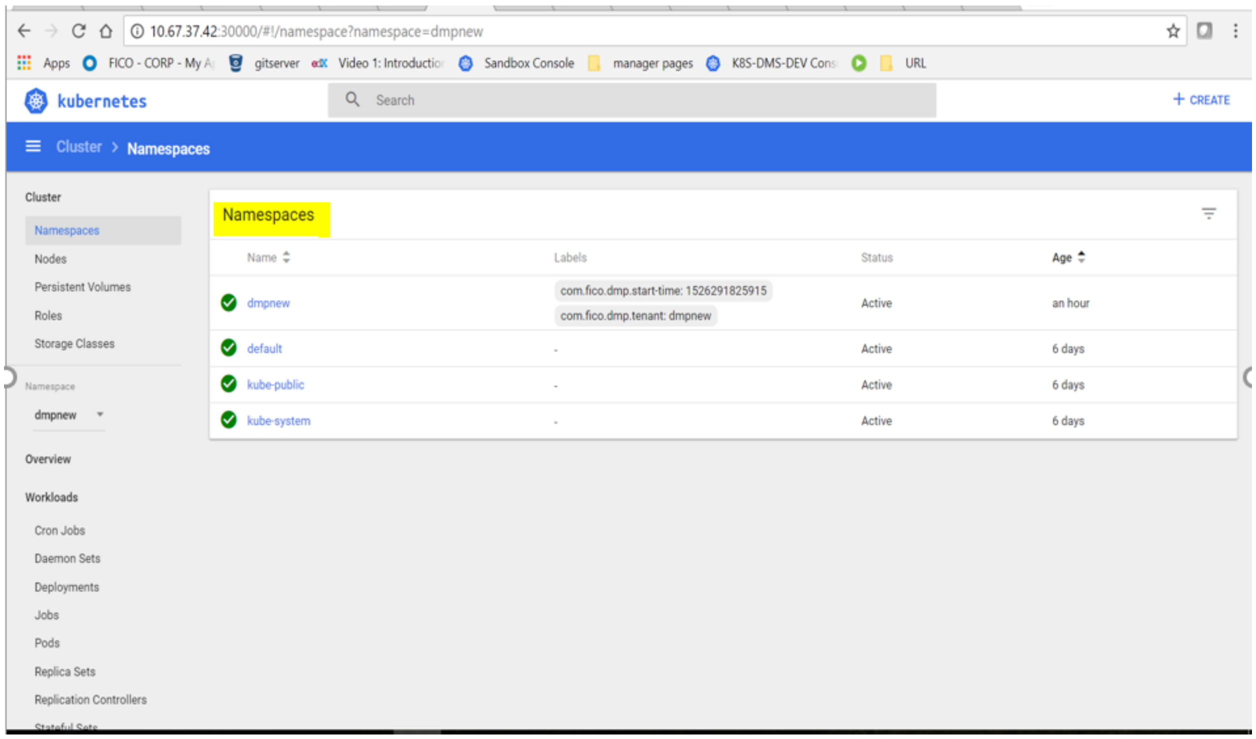
5. **ConfigMap and Secret**: ConfigMap makes it possible to inject a configuration based on an environment while keeping the container image identical across multiple environments. These can be injected by mounting volumes or environment variables, and it stores these values in the key/value format.Secrets are used to store sensitive data such as passwords, OAuth tokens, etc
6. **Rolling Deployment and Rollback:**A deployment object holds one or more replica sets to support the rollback mechanism. In other words, it creates a new replica set every time the deployment configuration is changed and keeps the previous version in order to have the option of rollback. Only one replica set will be in active state at a certain time.
7. **Deployment**:A readying controller provides declarative updates for Pods and ReplicaSets. You describe a desired state in a very readying object, and also the readying controller changes the particular state to the required state at a controlled rate. you'll outline Deployments to make new ReplicaSets, or to get rid of existing Deployments and adopt all their resources with new Deployments.
8. **Ingress**: An API object that manages external access to the services in an exceedingly cluster, generally communications protocol. Ingress will give load reconciliation, SSL termination and name-based virtual hosting.
9. **Configuration**: To avoid having to unnecessarily construct your instrumentality pictures, you ought to decouple your application's configuration information from the code needed to run it. There area unit a few ways in which of doing this, that you ought to select per your use case:

Table 4.1 Configuration data

Approach	Type of Data	How it's mounted	Example
Using a manifest's	Non-	Environment variable	Command-line

Approach	Type of Data	How it's mounted	Example
container definition	confidential		flag
Using ConfigMaps	Non-confidential	Environment variable OR local file	nginx configuration
Using Secrets	Confidential	Environment variable OR local file	Database credentials

DMP is deployed (using above technologies) i.e it first create a namespace, creates The entire process is then automated in the java class. So we need to run this class to run our test which will deploy the DMP. Once the test runs successfully we are able to see the DMP environment console on Kubernetes cluster. Now a namespace with name (dmpnew) given in the test is created which looks like this:

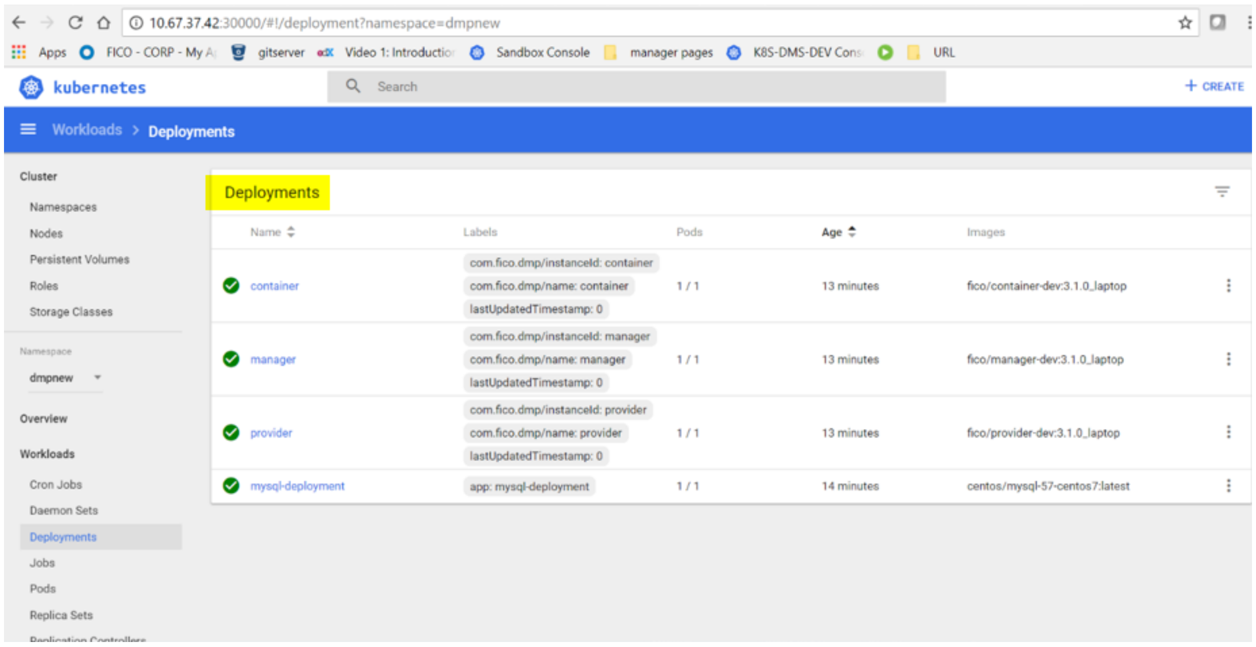


Namespaces

Name	Labels	Status	Age
✓ dmpnew	com.fico.dmp.start-time: 1526291825915 com.fico.dmp.tenant: dmpnew	Active	an hour
✓ default	-	Active	6 days
✓ kube-public	-	Active	6 days
✓ kube-system	-	Active	6 days

Figure 4.2 Namespaces

The test create four deployments Manager, Container, Provider and Mysql-deployment.

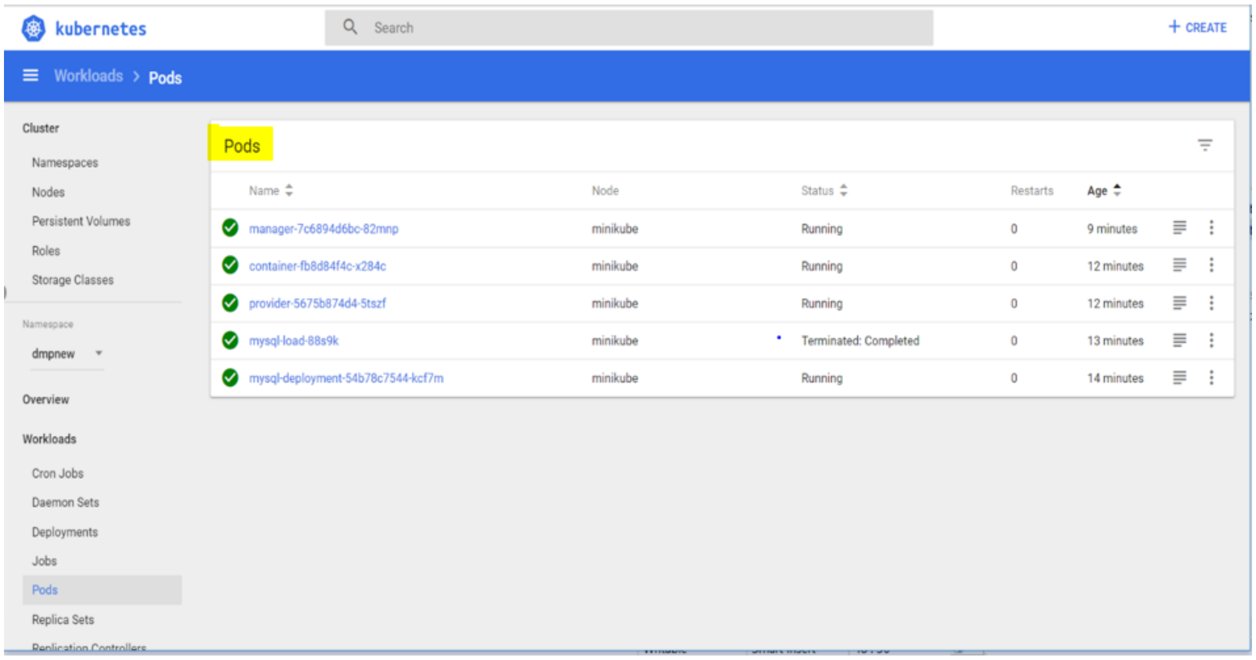


Deployments

Name	Labels	Pods	Age	Images
✓ container	com.fico.dmp/instanceId: container com.fico.dmp/name: container lastUpdatedTimestamp: 0	1 / 1	13 minutes	fico/container-dev:3.1.0_laptop
✓ manager	com.fico.dmp/instanceId: manager com.fico.dmp/name: manager lastUpdatedTimestamp: 0	1 / 1	13 minutes	fico/manager-dev:3.1.0_laptop
✓ provider	com.fico.dmp/instanceId: provider com.fico.dmp/name: provider lastUpdatedTimestamp: 0	1 / 1	13 minutes	fico/provider-dev:3.1.0_laptop
✓ mysql-deployment	app: mysql-deployment	1 / 1	14 minutes	centos/mysql-57-centos7:latest

Figure 4.3 Deployments

Corresponding to the above deployments four pods are created.

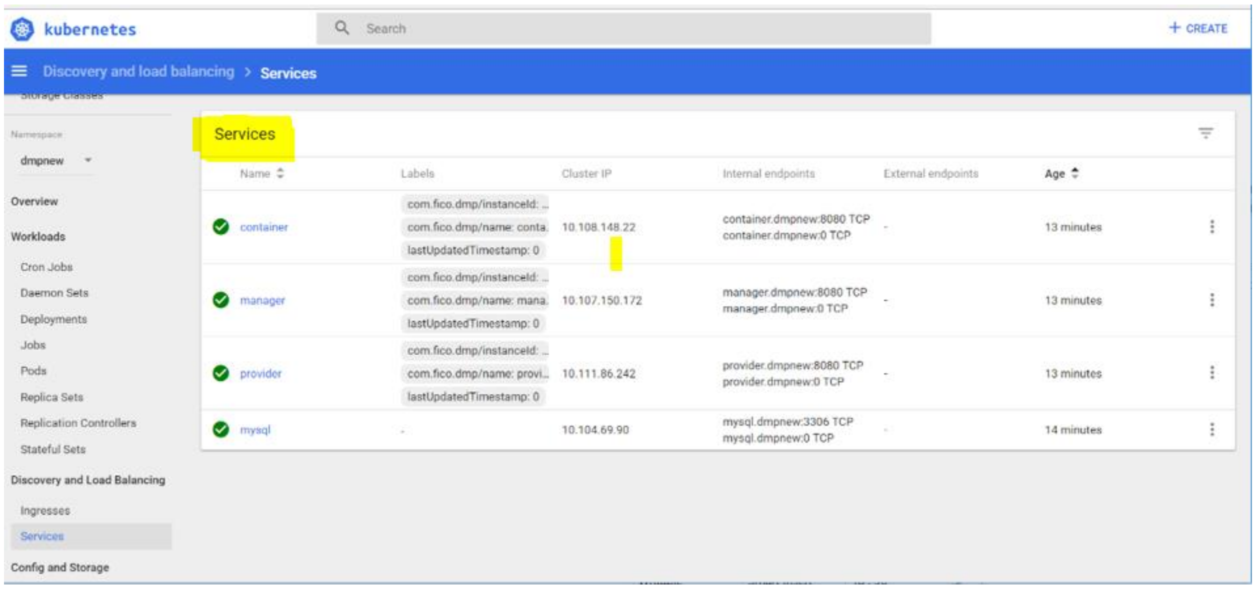


Pods

	Name	Node	Status	Restarts	Age
✓	manager-7c6894d6bc-82mnp	minikube	Running	0	9 minutes
✓	container-fb8d84f4c-x284c	minikube	Running	0	12 minutes
✓	provider-5675b874d4-5tszf	minikube	Running	0	12 minutes
✓	mysql-load-88s9k	minikube	Terminated: Completed	0	13 minutes
✓	mysql-deployment-54b78c7544-kcf7m	minikube	Running	0	14 minutes

Figure 4.4 Pods

Once the deployment and pods are created services are created, so our tests creates four services:

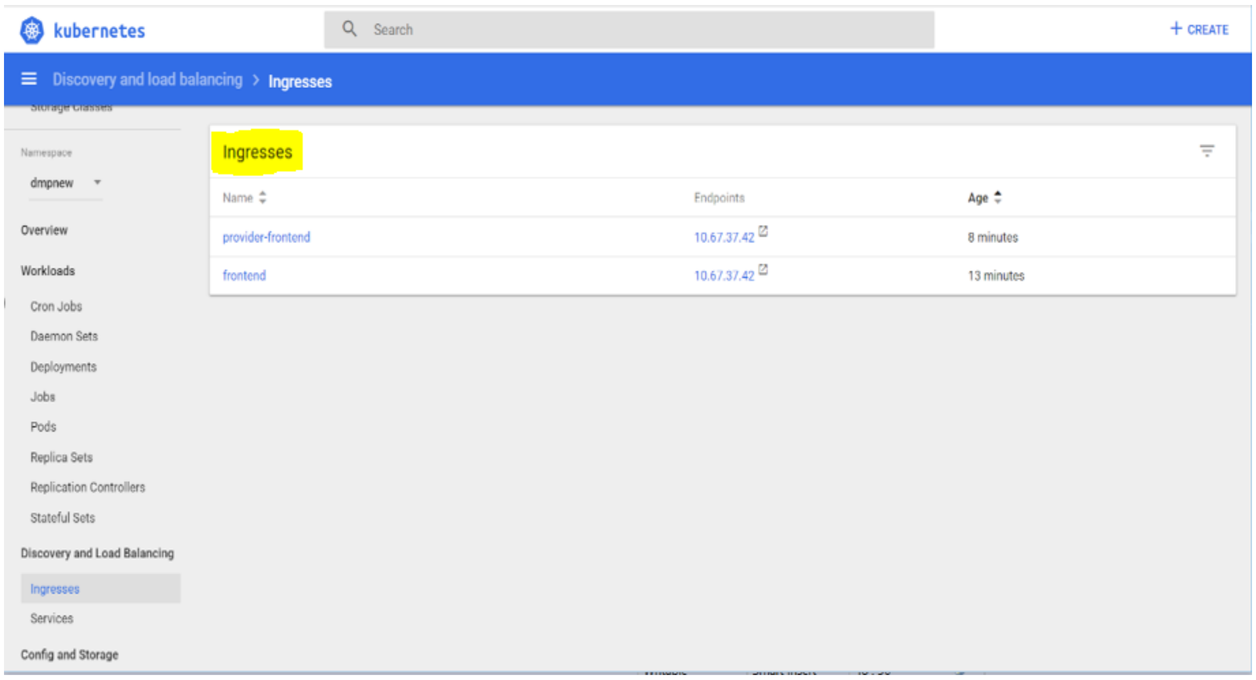


Services

	Name	Labels	Cluster IP	Internal endpoints	External endpoints	Age
✓	container	com.fico.dmp/instanceId: ... com.fico.dmp/name: conta. lastUpdatedTimestamp: 0	10.108.148.22	container.dmpnew:8080 TCP container.dmpnew:0 TCP	-	13 minutes
✓	manager	com.fico.dmp/instanceId: ... com.fico.dmp/name: mana. lastUpdatedTimestamp: 0	10.107.150.172	manager.dmpnew:8080 TCP manager.dmpnew:0 TCP	-	13 minutes
✓	provider	com.fico.dmp/instanceId: ... com.fico.dmp/name: provi... lastUpdatedTimestamp: 0	10.111.86.242	provider.dmpnew:8080 TCP provider.dmpnew:0 TCP	-	13 minutes
✓	mysql	-	10.104.69.90	mysql.dmpnew:3306 TCP mysql.dmpnew:0 TCP	-	14 minutes

Figure4.5 Services

Then ingress are created to manage the external(outside cluster) access to the services(that you have created for the pod) in a cluster, typically HTTP(s). Ingress can provide load balancing(external), SSL termination and name-based virtual hosting.

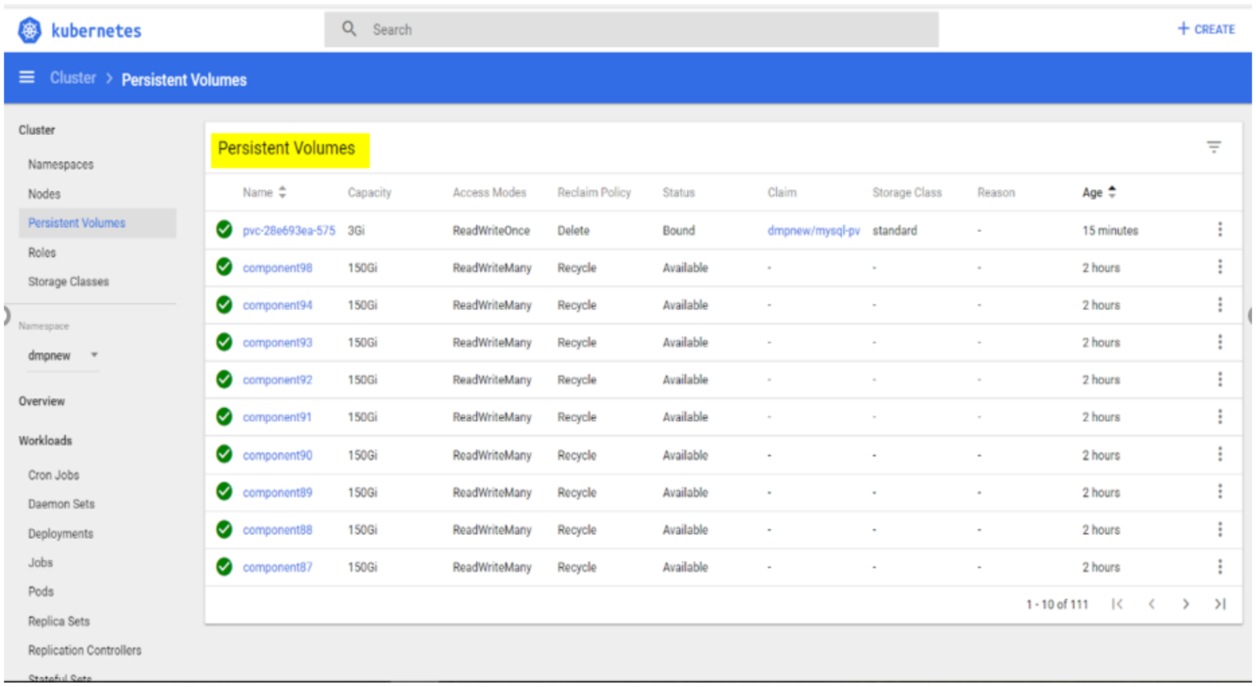


Ingresses

Name	Endpoints	Age
provider-frontend	10.67.37.42	8 minutes
frontend	10.67.37.42	13 minutes

Figure 4.6 Ingresses

For storage at the cluster level generally a cluster operator defines the PersistentVolume objects for the cluster, and cluster users (application developers) outline the PersistentVolumeClaim objects that your application requires:



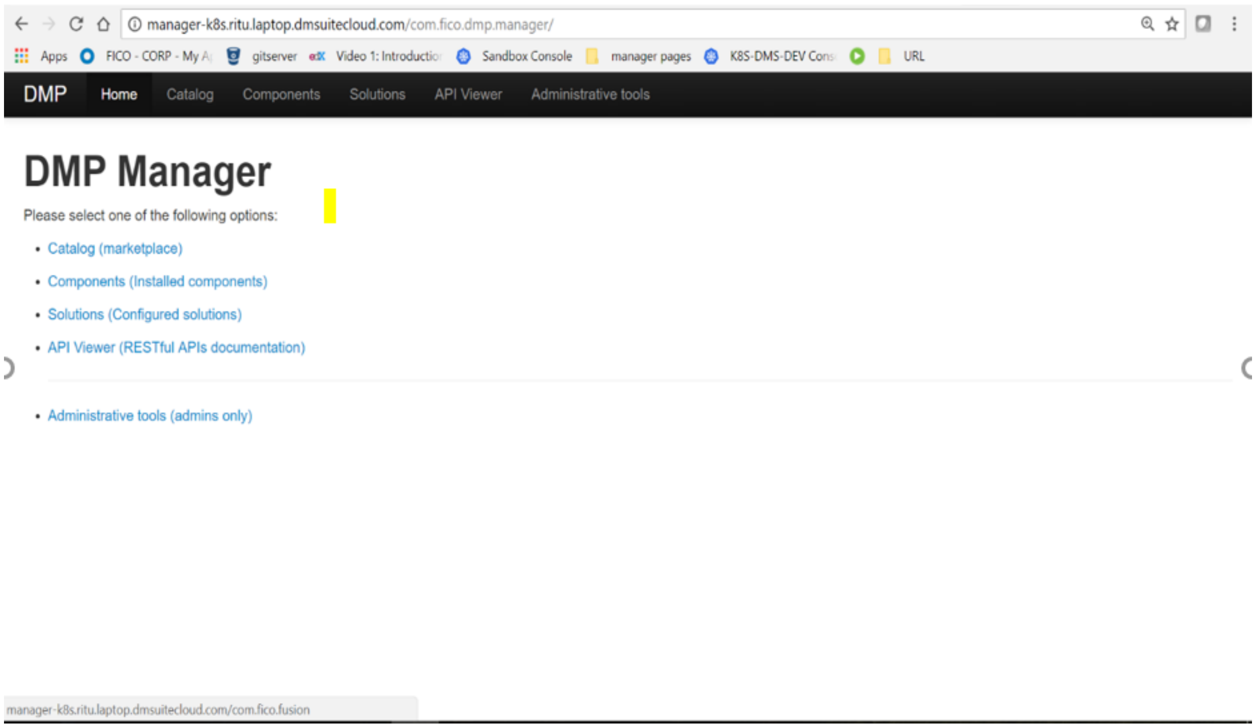
Persistent Volumes

Name	Capacity	Access Modes	Reclaim Policy	Status	Claim	Storage Class	Reason	Age
✓ pvc-28e693ea-575	3Gi	ReadWriteOnce	Delete	Bound	dmpnew/mysql-pv	standard	-	15 minutes
✓ component98	150Gi	ReadWriteMany	Recycle	Available	-	-	-	2 hours
✓ component94	150Gi	ReadWriteMany	Recycle	Available	-	-	-	2 hours
✓ component93	150Gi	ReadWriteMany	Recycle	Available	-	-	-	2 hours
✓ component92	150Gi	ReadWriteMany	Recycle	Available	-	-	-	2 hours
✓ component91	150Gi	ReadWriteMany	Recycle	Available	-	-	-	2 hours
✓ component90	150Gi	ReadWriteMany	Recycle	Available	-	-	-	2 hours
✓ component89	150Gi	ReadWriteMany	Recycle	Available	-	-	-	2 hours
✓ component88	150Gi	ReadWriteMany	Recycle	Available	-	-	-	2 hours
✓ component87	150Gi	ReadWriteMany	Recycle	Available	-	-	-	2 hours

1 - 10 of 111

Figure 4.7 Persistent Volumes

After all this is done then by using the external name defined in the test class you are able to see the Manager page which is destined for the component developers and other internal teams:



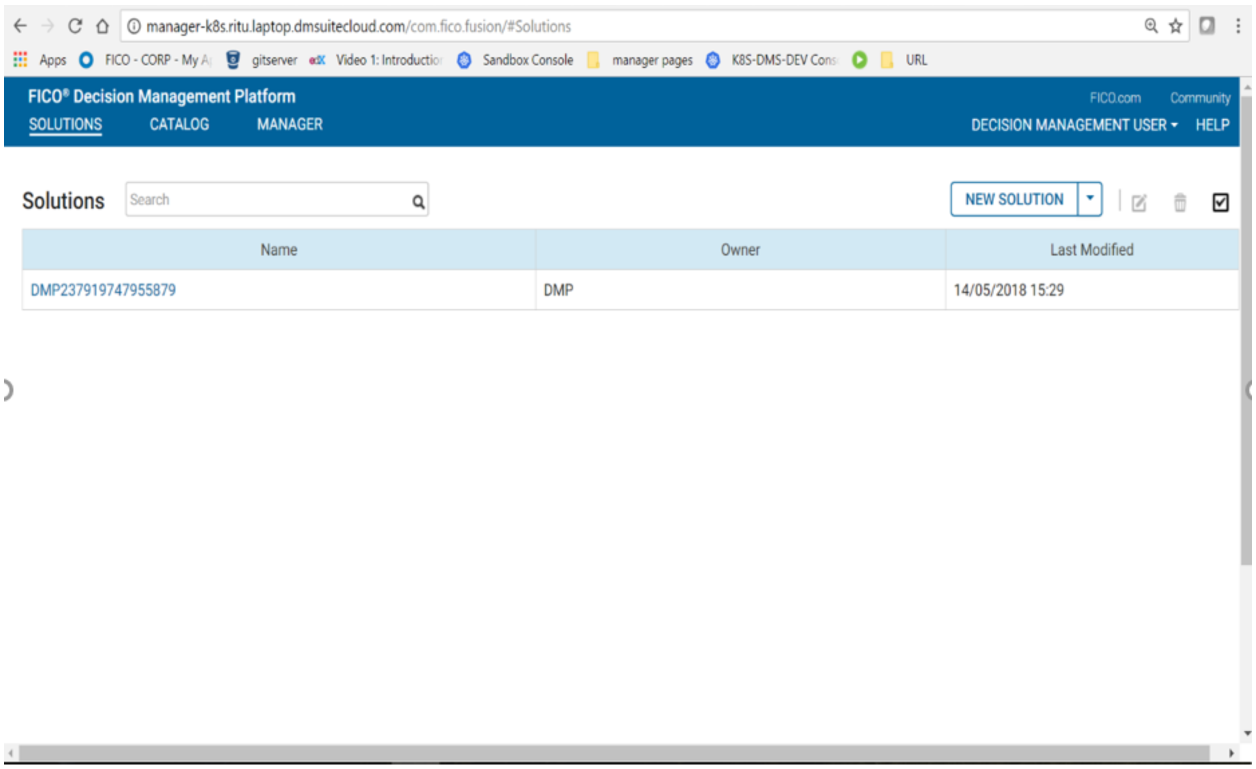
DMP Manager

Please select one of the following options:

- Catalog (marketplace)
- Components (Installed components)
- Solutions (Configured solutions)
- API Viewer (RESTful APIs documentation)

- Administrative tools (admins only)

For users there is separate page which is known as Fusion page where users can create component, performs LCM on the component and execute services etc.


manager-k8s.ritu.laptop.dmsuitecloud.com/com.fico.fusion/#Solutions
FICO® Decision Management Platform
SOLUTIONS
CATALOG
MANAGER
FICO.com
Community
DECISION MANAGEMENT USER
HELP
Solutions
Search
NEW SOLUTION

Name	Owner	Last Modified
DMP237919747955879	DMP	14/05/2018 15:29

Figure 4.8 DMP Manager Page

In the Fusion page a solution is created as defined in the introduction part:

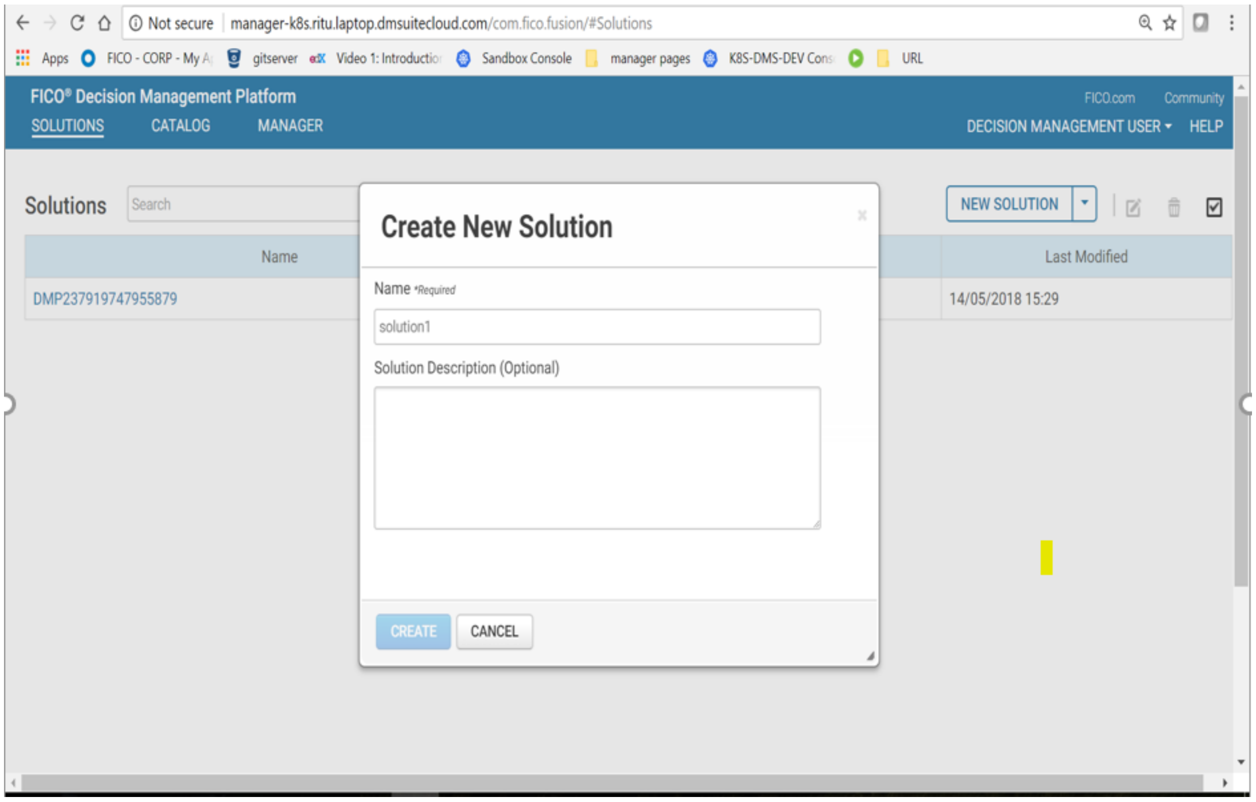
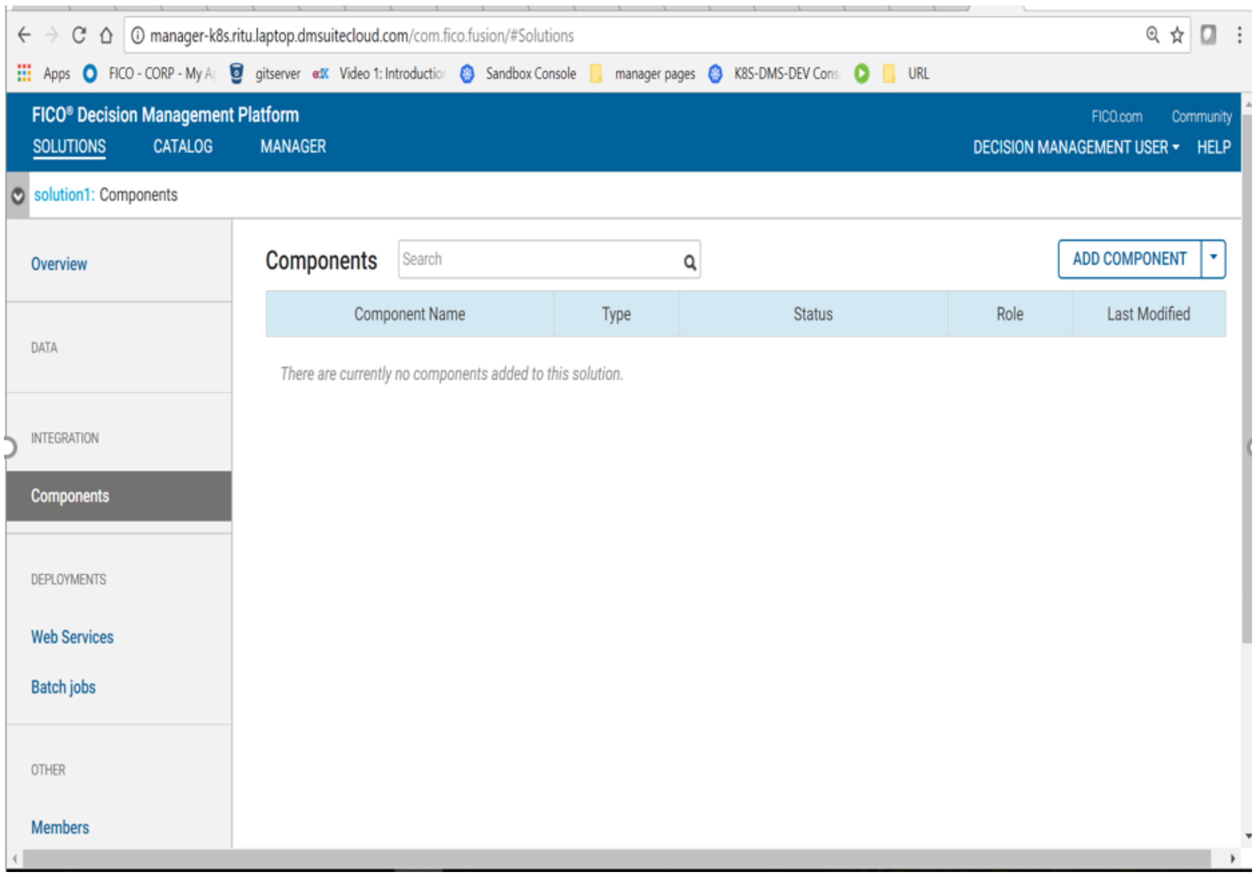


Figure 4.9 Solution Creation

Once the solution is created then you can test component lifecycle in it:



Component Name	Type	Status	Role	Last Modified

There are currently no components added to this solution.

Figure 4.10 Component Creation page

Chapter 5

DockerizedComponents

During internship period, a task to dockerize various components to move from openshift based platform to Kubernetes based platform is given to me. Some of these components are:

File Exchange ComponentOverview

The File Exchange Component allows a solution user to have files created by the solution automatically transferred to a target SFTP server based upon entered configuration information.

Configuration Form

The Configuration Form allows users to associate SFTP connection information with a DMP solution. This information is used to download files from a target source location (per the Source Path field) to the specified destination (per the Destination Path field) when a file creation event is detected in the source location.

Once the SFTP configuration information is entered, the user generates a public/private key pair via the Generate Key button. The user downloads the public key for installation in the target SFTP server's SSH configuration file. It is used to authenticate/authorize subsequent download requests.

Separate SFTP configuration information must be entered for each life-cycle state - i.e., Development, Staging, or Production - by selecting the appropriate tab.

Event History

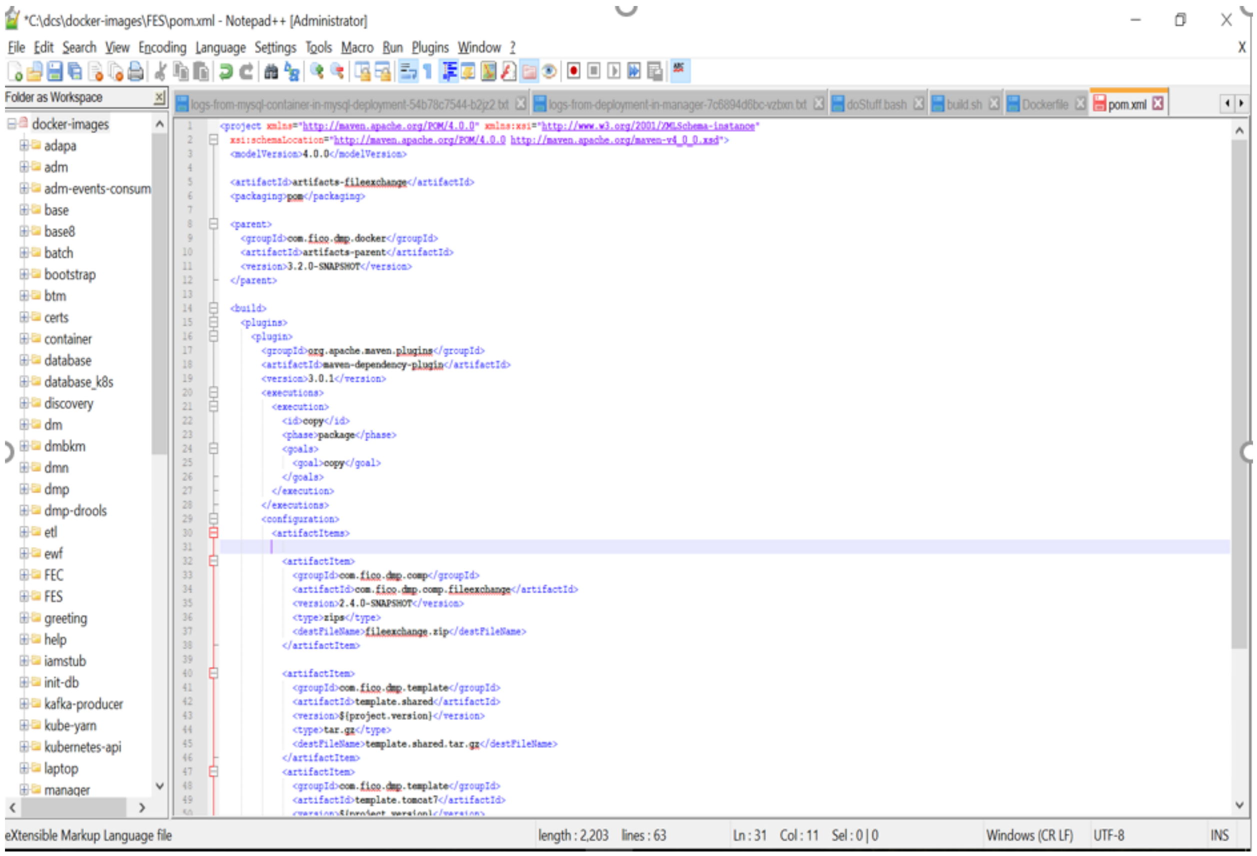
Event History provides a tabular view of in-process or past file download attempts. The table entries are partitioned by life-cycle state - i.e., Development, Staging, or Production - based upon which tab is selected. The entries can be further filtered by operation status:

- READY
- IN_PROGRESS
- COMPLETED
- FAILED
- FAILED_RETRY

I helped in dockerizing the FEC component and deploying it on the DMP environment.To dockerize the FEC component I followed the following steps:

Pom.xml: To dockerize any component usually three files are required pom.xml, build.sh, and Dockerfile depending upon the requirement of application.These files are described in detail below:

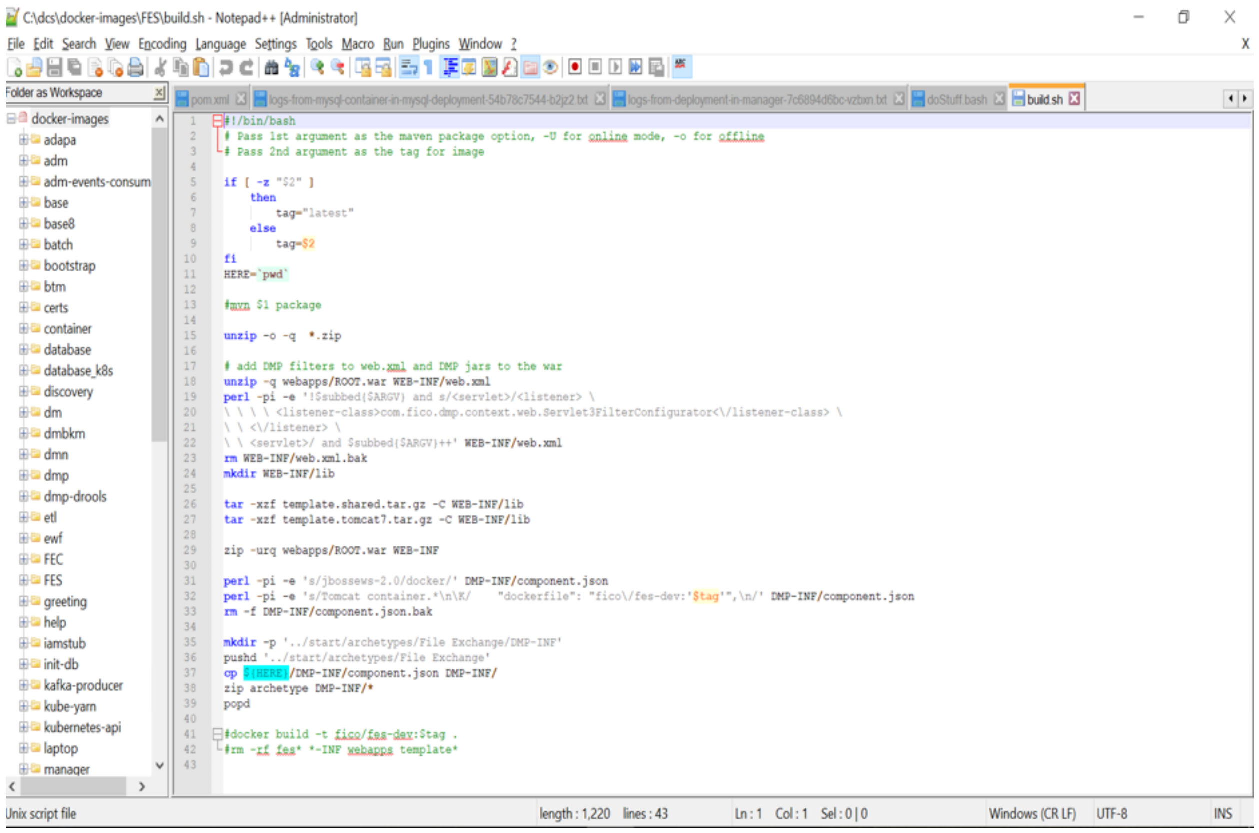
A POM.xml provides all the artifacts required for a project. This pom is required to pull all the artifacts required for the application and which needs to be packaged with docker image. For example, pom.xml file for FEC looks like this:



```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <artifactId>artifacts-fileexchange</artifactId>
  <packaging>pom</packaging>

  <parent>
    <groupId>com.fico.dmp.docker</groupId>
    <artifactId>artifacts-parent</artifactId>
    <version>3.2.0-SNAPSHOT</version>
  </parent>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-dependency-plugin</artifactId>
        <version>3.0.1</version>
        <executions>
          <execution>
            <id>copy</id>
            <phase>package</phase>
            <goals>
              <goal>copy</goal>
            </goals>
          </execution>
        </executions>
        <configuration>
          <artifactItems>

            <artifactItem>
              <groupId>com.fico.dmp.comp</groupId>
              <artifactId>com.fico.dmp.comp.fileexchange</artifactId>
              <version>2.4.0-SNAPSHOT</version>
              <type>zips</type>
              <destFileName>fileexchange.zip</destFileName>
            </artifactItem>

            <artifactItem>
              <groupId>com.fico.dmp.template</groupId>
              <artifactId>template.shared</artifactId>
              <version>${project.version}</version>
              <type>tar.gz</type>
              <destFileName>template.shared.tar.gz</destFileName>
            </artifactItem>
            <artifactItem>
              <groupId>com.fico.dmp.template</groupId>
              <artifactId>template.tomcat7</artifactId>
              <version>${project.version}</version>
```

Figure 5.1 Pom File for FEC

Build.sh:- This is a shell script file which contains the logic that carries out your build steps and builds the docker image for your application. A snapshot of FEC docker image is attached below:



```bash
#!/bin/bash
# Pass 1st argument as the maven package option, -U for online mode, -o for offline
# Pass 2nd argument as the tag for image

if [ -z "$2" ]
    then
        tag="latest"
    else
        tag=$2
fi
HERE=`pwd`

#mvn $1 package

unzip -o -q  *.zip

# add DMP filters to web.xml and DMP jars to the war
unzip -q webapps/ROOT.war WEB-INF/web.xml
perl -pi -e '!$subbed{$ARGV} and s/<servlet>/<listener> \
\ \ \ \ <listener-class>com.fico.dmp.context.web.Servlet3FilterConfigurator<\/listener-class> \
\ \ <\/listener> \
\ \ <servlet>/ and $subbed{$ARGV}++' WEB-INF/web.xml
rm WEB-INF/web.xml.bak
mkdir WEB-INF/lib

tar -xzf template.shared.tar.gz -C WEB-INF/lib
tar -xzf template.tomcat7.tar.gz -C WEB-INF/lib

zip -urq webapps/ROOT.war WEB-INF

perl -pi -e 's/jbossews-2.0/docker/' DMP-INF/component.json
perl -pi -e 's/Tomcat container.*\n\K/    "dockerfile": "fico\/fes-dev:'$tag'",\n/' DMP-INF/component.json
rm -f DMP-INF/component.json.bak

mkdir -p '../start/archetypes/File Exchange/DMP-INF'
pushd '../start/archetypes/File Exchange'
cp ${HERE}/DMP-INF/component.json DMP-INF/
zip archetype DMP-INF/*
popd

#docker build -t fico/fes-dev:$tag .
#rm -rf fes* *-INF webapps template*
```

Figure 5.2 Build File For FEC

DockerFile: - A **Dockerfile** is a text document (with no extension) that contains all the commands a user could call on the command line to assemble an image.

1. Provide the base image from which you want to create your image. Standard docker, Google and Red Hat base image may beused, provided they do not require separate licensing.

2. Specify the user with which you want to setup your docker image i.e as root or as a different user.

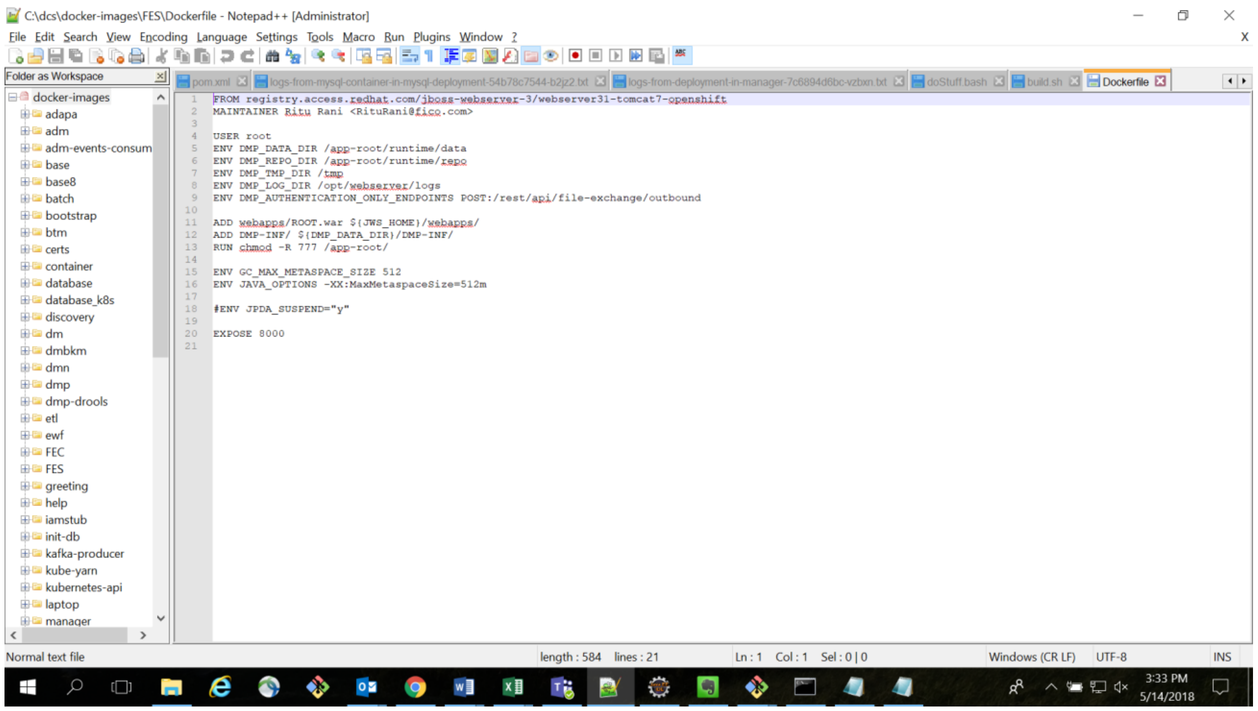
3. Specify all the environment variables used by your component like DMP_DATA_DIR, DMP_REPO_DIR etc

4. ADD your application war to server webapps folder and DMP-INF to DMP_DATA_DIR.

5. Give access rights to files and folders if required using chmod command.

6. [optional] provide the GC_MAX_METASPACE_SIZE.

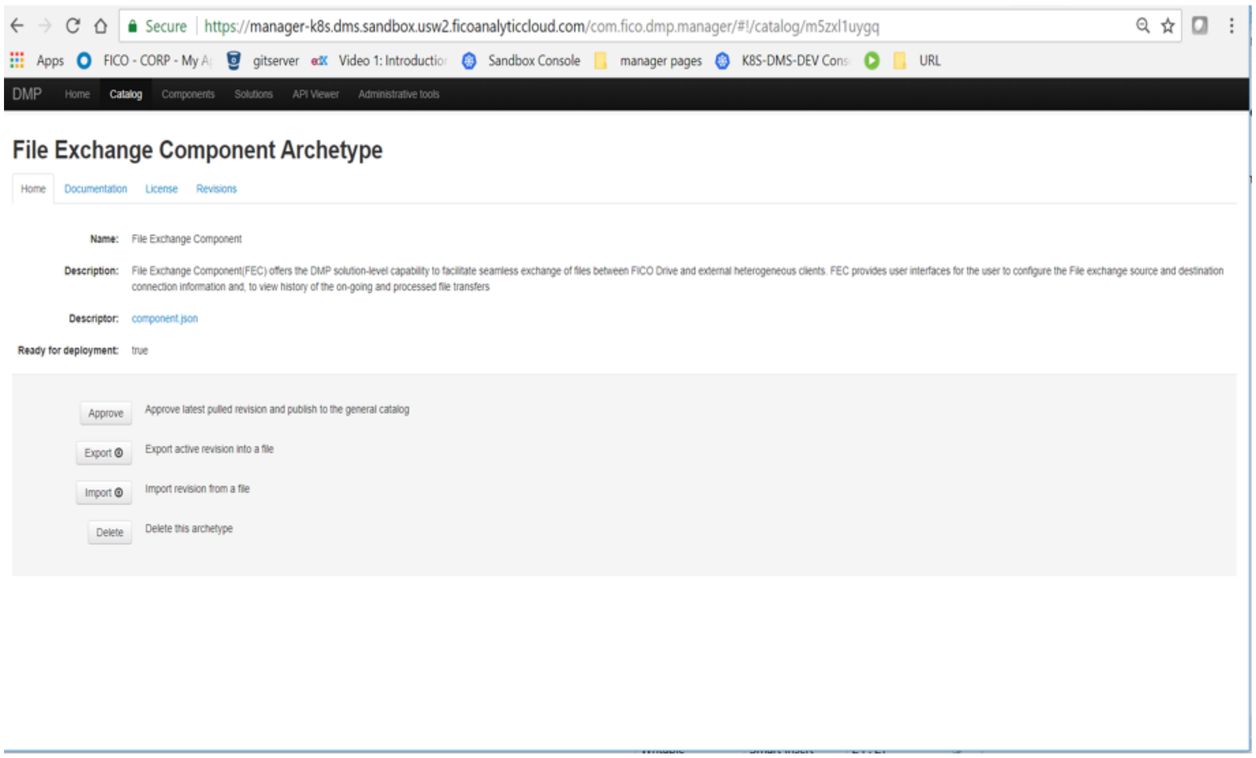
7. Expose PORTs that your application uses.



```
FROM registry.access.redhat.com/jboss-webserver-3/webserver31-tomcat7-openshift
MAINTAINER Ritu Rani <RituRani@fico.com>

USER root
ENV DMP_DATA_DIR /app-root/runtime/data
ENV DMP_REPO_DIR /app-root/runtime/repo
ENV DMP_TMP_DIR /tmp
ENV DMP_LOG_DIR /opt/webserver/logs
ENV DMP_AUTHENTICATION_ONLY_ENDPOINTS POST:/rest/api/file-exchange/outbound

ADD webapps/ROOT.war ${JWS_HOME}/webapps/
ADD DMP-INF/ ${DMP_DATA_DIR}/DMP-INF/
RUN chmod -R 777 /app-root/

ENV GC_MAX_METASPACE_SIZE 512
ENV JAVA_OPTIONS -XX:MaxMetaspaceSize=512m

#ENV JPDA_SUSPEND="y"

EXPOSE 8000
```

Figure 5.3DockerFileFor FEC

- Build the docker-image: Execute build.sh command to build the docker image
- Tag the image with the desired tag.
- Push the image to Docker Hub
- Create an archetype in DMP environment:


https://manager-k8s.dms.sandbox.usw2.ficoanalyticcloud.com/com.fico.dmp.manager/#!/catalog/m5zxl1uygq
DMP Home Catalog Components Solutions API Viewer Administrative tools
Home Documentation License Revisions
Approve
Export
Import
Delete

File Exchange Component Archetype

Name: File Exchange Component

Description: File Exchange Component(FEC) offers the DMP solution-level capability to facilitate seamless exchange of files between FICO Drive and external heterogeneous clients. FEC provides user interfaces for the user to configure the File exchange source and destination connection information and, to view history of the on-going and processed file transfers

Descriptor: component.json

Ready for deployment: true

Approve latest pulled revision and publish to the general catalog

Export active revision into a file

Import revision from a file

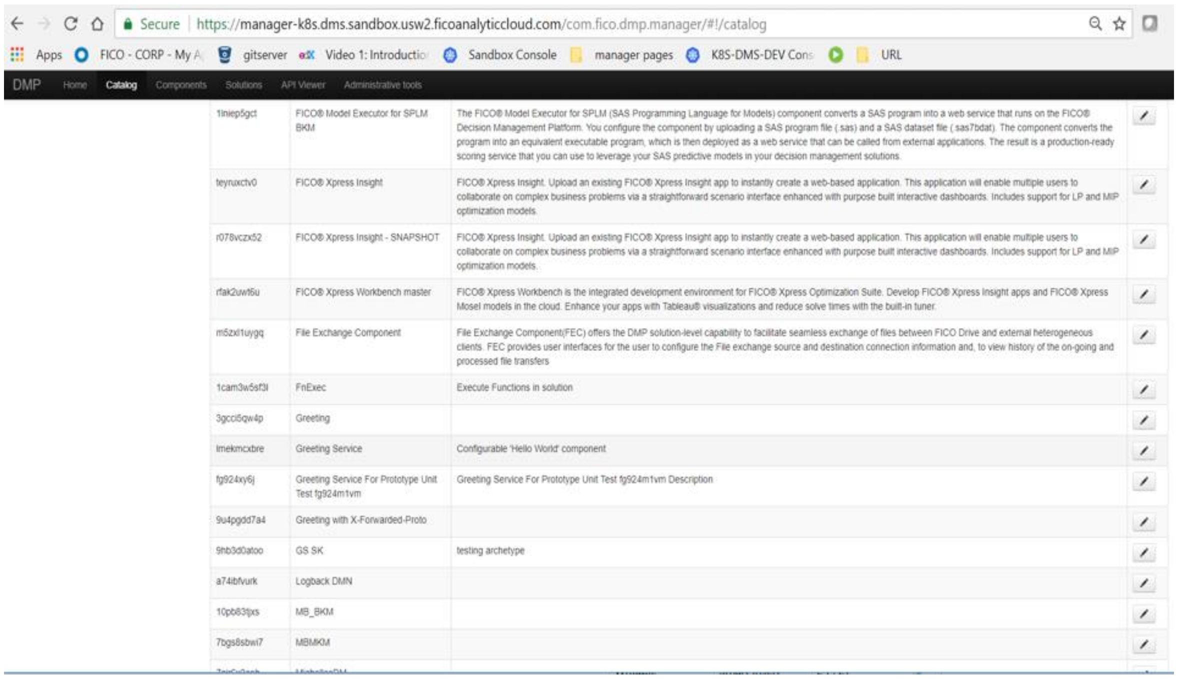
Delete this archetype

Figure 5.4 FEC Archetype

- Upload the archetype zip file in implementation "Choose File" button.

4. If Archetype is already present or want to upload new version of archetype
    - click on the Edit button of that archetype
    - import that zip file
    - Approve the latest revision of the archetype

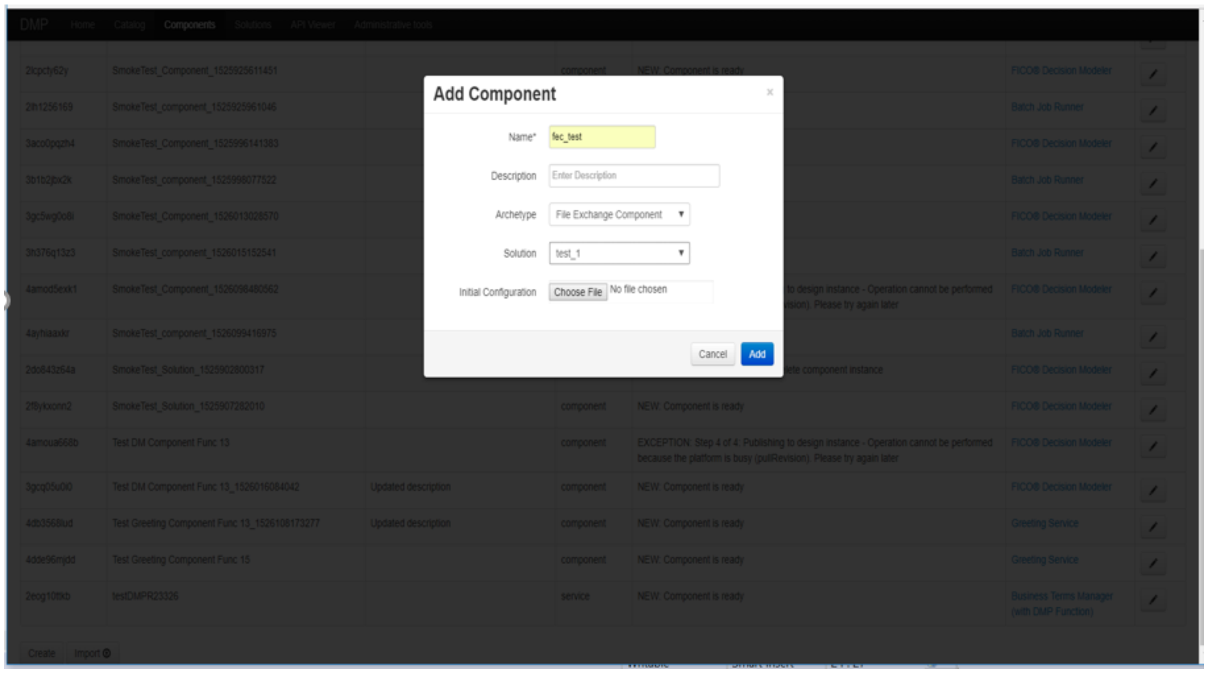
Now FEC prototype looks like this in DMP Catalog:



https://manager-k8s.dms.sandbox.usw2.ficoanalyticcloud.com/com.fico.dmp.manager/#!/catalog

1lniep5gct	FICO® Model Executor for SPLM BKM	The FICO® Model Executor for SPLM (SAS Programming Language for Models) component converts a SAS program into a web service that runs on the FICO® Decision Management Platform. You configure the component by uploading a SAS program file (.sas) and a SAS dataset file (.sas7bdat). The component converts the program into an equivalent executable program, which is then deployed as a web service that can be called from external applications. The result is a production-ready scoring service that you can use to leverage your SAS predictive models in your decision management solutions.
teyruxctv0	FICO® Xpress Insight	FICO® Xpress Insight. Upload an existing FICO® Xpress Insight app to instantly create a web-based application. This application will enable multiple users to collaborate on complex business problems via a straightforward scenario interface enhanced with purpose built interactive dashboards. Includes support for LP and MIP optimization models.
r078vczx52	FICO® Xpress Insight - SNAPSHOT	FICO® Xpress Insight. Upload an existing FICO® Xpress Insight app to instantly create a web-based application. This application will enable multiple users to collaborate on complex business problems via a straightforward scenario interface enhanced with purpose built interactive dashboards. Includes support for LP and MIP optimization models.
rfak2uwt6u	FICO® Xpress Workbench master	FICO® Xpress Workbench is the integrated development environment for FICO® Xpress Optimization Suite. Develop FICO® Xpress Insight apps and FICO® Xpress Mosel models in the cloud. Enhance your apps with Tableau® visualizations and reduce solve times with the built-in tuner.
m5zxl1uygq	File Exchange Component	File Exchange Component(FEC) offers the DMP solution-level capability to facilitate seamless exchange of files between FICO Drive and external heterogeneous clients. FEC provides user interfaces for the user to configure the File exchange source and destination connection information and, to view history of the on-going and processed file transfers
1cam3w5sf3l	FnExec	Execute Functions in solution
3gcci5qw4p	Greeting	
lmekmcxbre	Greeting Service	Configurable 'Hello World' component
fg924xy6j	Greeting Service For Prototype Unit Test fg924m1vm	Greeting Service For Prototype Unit Test fg924m1vm Description
9u4pgdd7a4	Greeting with X-Forwarded-Proto	
9hb3d0atoo	GS SK	testing archetype
a74ibfvurk	Logback DMN	
10pb83tjxs	MB_BKM	
7bgs8sbwi7	MBMKM	

Figure 5.5 FEC Archetype in Catalog



Add Component

Name*: fec_test

Description: Enter Description

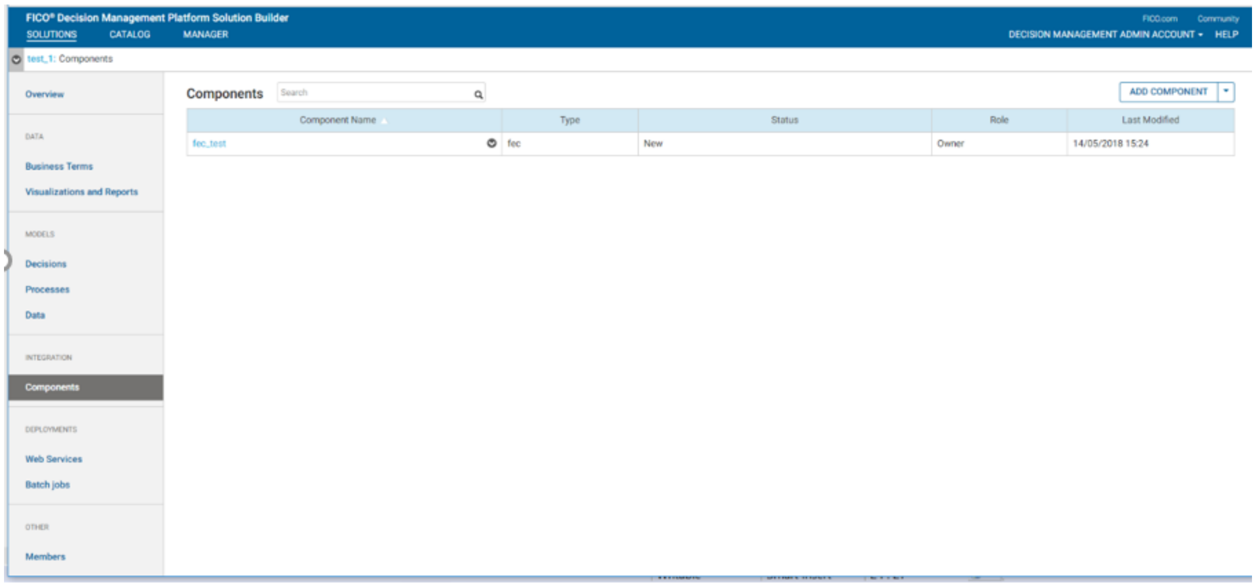
Archetype: File Exchange Component

Solution: test_1

Initial Configuration: Choose File No file chosen

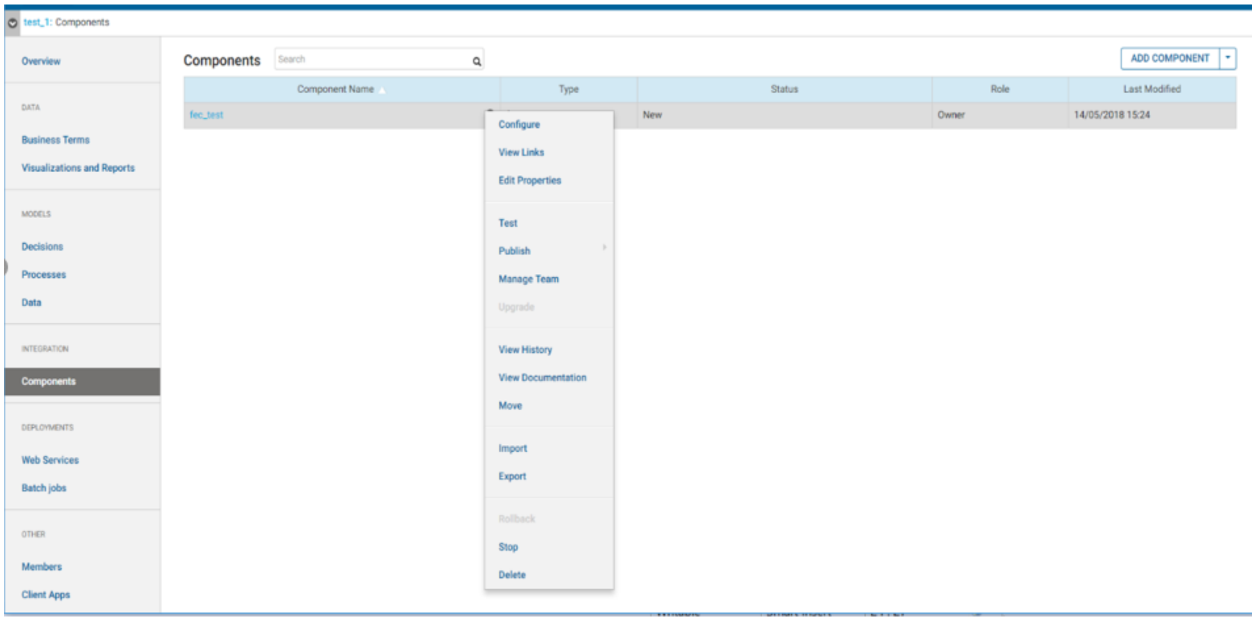
2lcpcty62y	SmokeTest_Component_1525925611451		component	NEW: Component is ready	FICO® Decision Modeler
2lh1256169	SmokeTest_component_1525925961046				Batch Job Runner
3aco0pqzh4	SmokeTest_Component_1525996141383				FICO® Decision Modeler
3b1b2jbx2k	SmokeTest_component_1525998077522				Batch Job Runner
3gc5wg0o8i	SmokeTest_Component_1526013028570				FICO® Decision Modeler
3h376q13z3	SmokeTest_component_1526015152541				Batch Job Runner
4amod5exk1	SmokeTest_Component_1526098480562			to design instance - Operation cannot be performed vision). Please try again later	FICO® Decision Modeler
4ayhiaaxkr	SmokeTest_component_1526099416975				Batch Job Runner
2do843z64a	SmokeTest_Solution_1525902800317			lete component instance	FICO® Decision Modeler
2f8ykxonn2	SmokeTest_Solution_1525907282010		component	NEW: Component is ready	FICO® Decision Modeler
4amoua668b	Test DM Component Func 13		component	EXCEPTION: Step 4 of 4: Publishing to design instance - Operation cannot be performed because the platform is busy (pullRevision). Please try again later	FICO® Decision Modeler
3gcq05u0i0	Test DM Component Func 13_1526016084042	Updated description	component	NEW: Component is ready	FICO® Decision Modeler
4db3568iud	Test Greeting Component Func 13_1526108173277	Updated description	component	NEW: Component is ready	Greeting Service
4dde96mjdd	Test Greeting Component Func 15		component	NEW: Component is ready	Greeting Service
2eog10ttkb	testDMPR23326		service	NEW: Component is ready	Business Terms Manager (with DMP Function)

Figure 5.6 FEC Component Creation



Component Name	Type	Status	Role	Last Modified
fec_test	fec	New	Owner	14/05/2018 15:24

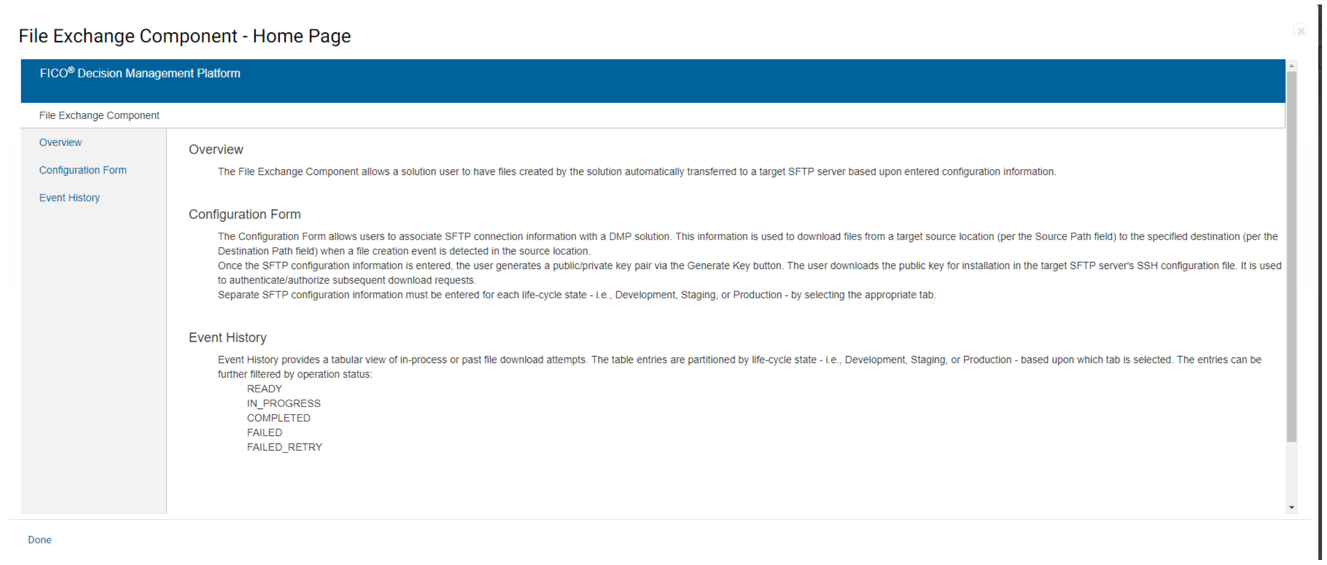
Figure 5.7 FEC component in New state



Component Name	Type	Status	Role	Last Modified
fec_test		New	Owner	14/05/2018 15:24

Figure 5.8 Configuring FEC

Landing Page (Overview)


File Exchange Component - Home Page

FICO® Decision Management Platform

File Exchange Component

Overview

The File Exchange Component allows a solution user to have files created by the solution automatically transferred to a target SFTP server based upon entered configuration information.

Configuration Form

The Configuration Form allows users to associate SFTP connection information with a DMP solution. This information is used to download files from a target source location (per the Source Path field) to the specified destination (per the Destination Path field) when a file creation event is detected in the source location.
Once the SFTP configuration information is entered, the user generates a public/private key pair via the Generate Key button. The user downloads the public key for installation in the target SFTP server's SSH configuration file. It is used to authenticate/authorize subsequent download requests.
Separate SFTP configuration information must be entered for each life-cycle state - i.e., Development, Staging, or Production - by selecting the appropriate tab.

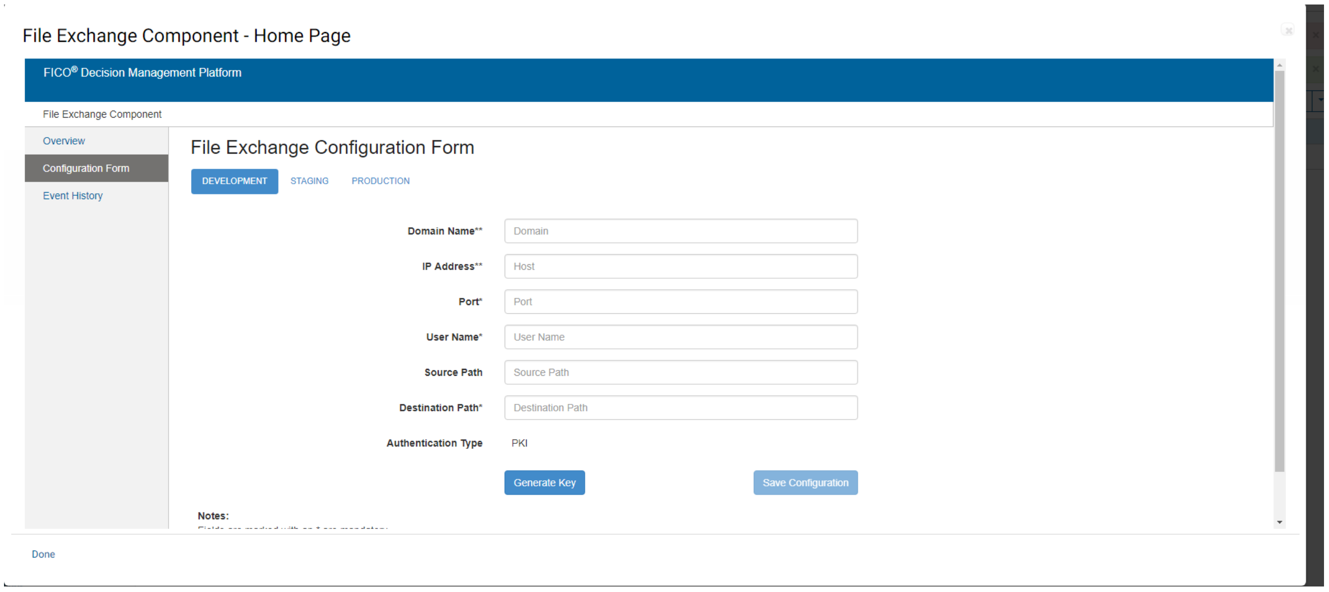
Event History

Event History provides a tabular view of in-process or past file download attempts. The table entries are partitioned by life-cycle state - i.e., Development, Staging, or Production - based upon which tab is selected. The entries can be further filtered by operation status:

- READY
- IN_PROGRESS
- COMPLETED
- FAILED
- FAILED_RETRY

Figure 5.9 Landing Page for File Exchange Component

Configuration Form Page


File Exchange Component - Home Page

FICO® Decision Management Platform

File Exchange Component

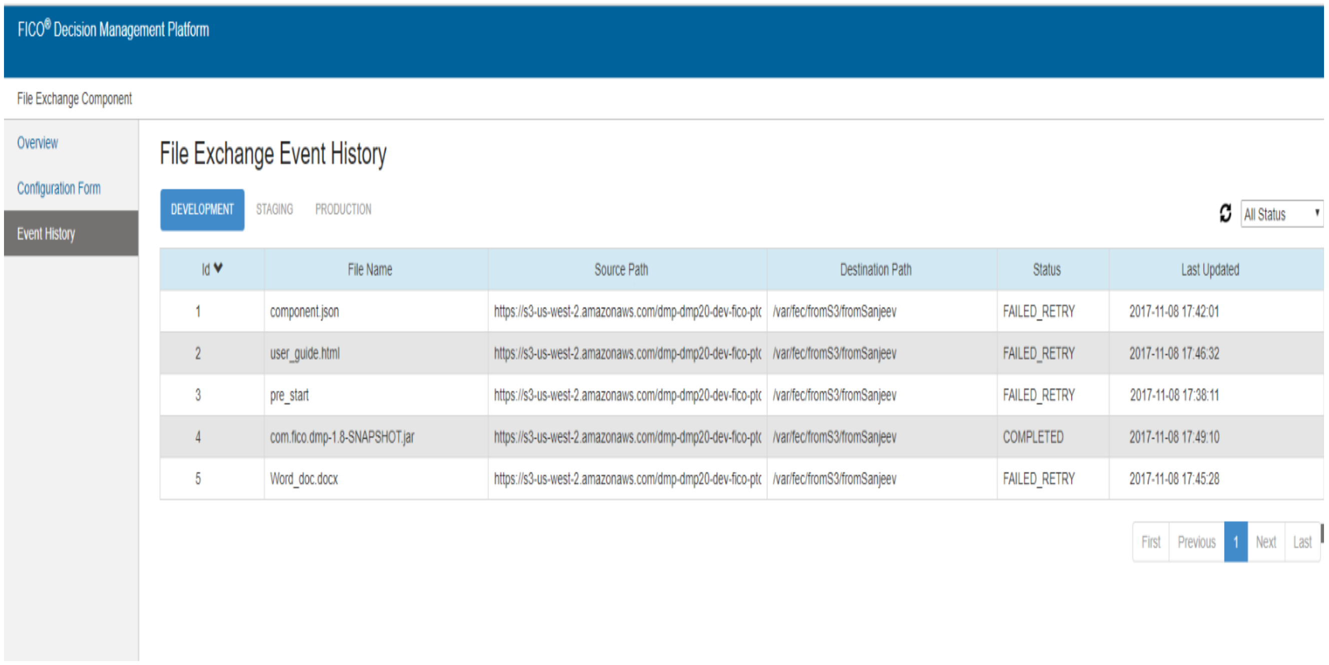
File Exchange Configuration Form

Domain Name**	Domain
IP Address**	Host
Port*	Port
User Name*	User Name
Source Path	Source Path
Destination Path*	Destination Path
Authentication Type	PKI

Notes:

Figure 5.10 Configuration Form page

Event History:



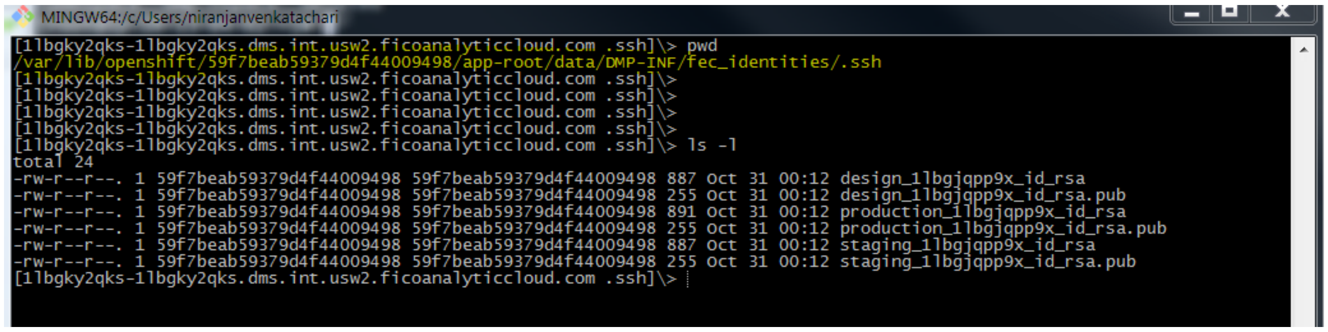
Id	File Name	Source Path	Destination Path	Status	Last Updated
1	component.json	https://s3-us-west-2.amazonaws.com/dmp-dmp20-dev-fico-ptc	/var/fec/fromS3/fromSanjeev	FAILED_RETRY	2017-11-08 17:42:01
2	user_guide.html	https://s3-us-west-2.amazonaws.com/dmp-dmp20-dev-fico-ptc	/var/fec/fromS3/fromSanjeev	FAILED_RETRY	2017-11-08 17:46:32
3	pre_start	https://s3-us-west-2.amazonaws.com/dmp-dmp20-dev-fico-ptc	/var/fec/fromS3/fromSanjeev	FAILED_RETRY	2017-11-08 17:38:11
4	com.fico.dmp-1.8-SNAPSHOT.jar	https://s3-us-west-2.amazonaws.com/dmp-dmp20-dev-fico-ptc	/var/fec/fromS3/fromSanjeev	COMPLETED	2017-11-08 17:49:10
5	Word_doc.docx	https://s3-us-west-2.amazonaws.com/dmp-dmp20-dev-fico-ptc	/var/fec/fromS3/fromSanjeev	FAILED_RETRY	2017-11-08 17:45:28

Figure 5.11 File Exchange Event History page

FEC Life-Cycle Management (LCM)

The FEC component allows the user to be able to save the file exchange configuration data, to be usable across all the 3 lifecycle environments. The FEC Configuration GUI is, however, available only in design environment. File Exchange Event history GUI is available for all 3 lifecycle environments.

In a typical work-flow, the FEC user generates keys (public and private) and saves the configuration data for each of the 3 life-cycle environments. The file ***feConfig.json (referred above)*** will contain 3 JSON configuration objects - one for each of the 3 environments, and the public and private key are also stored separated for each environment. See Fig. 5.4(a) below for sample listing of the files.



```
[1lbgky2qks-1lbgky2qks.dms.int.usw2.ficoanalyticcloud.com .ssh]\> pwd
/var/lib/openshift/59f7beab59379d4f44009498/app-root/data/DMP-INF/fec_identities/.ssh
[1lbgky2qks-1lbgky2qks.dms.int.usw2.ficoanalyticcloud.com .ssh]\>
[1lbgky2qks-1lbgky2qks.dms.int.usw2.ficoanalyticcloud.com .ssh]\>
[1lbgky2qks-1lbgky2qks.dms.int.usw2.ficoanalyticcloud.com .ssh]\>
[1lbgky2qks-1lbgky2qks.dms.int.usw2.ficoanalyticcloud.com .ssh]\>
[1lbgky2qks-1lbgky2qks.dms.int.usw2.ficoanalyticcloud.com .ssh]\> ls -l
total 24
-rw-r--r--. 1 59f7beab59379d4f44009498 59f7beab59379d4f44009498 887 Oct 31 00:12 design_1lbgjqpp9x_id_rsa
-rw-r--r--. 1 59f7beab59379d4f44009498 59f7beab59379d4f44009498 255 Oct 31 00:12 design_1lbgjqpp9x_id_rsa.pub
-rw-r--r--. 1 59f7beab59379d4f44009498 59f7beab59379d4f44009498 891 Oct 31 00:12 production_1lbgjqpp9x_id_rsa
-rw-r--r--. 1 59f7beab59379d4f44009498 59f7beab59379d4f44009498 255 Oct 31 00:12 production_1lbgjqpp9x_id_rsa.pub
-rw-r--r--. 1 59f7beab59379d4f44009498 59f7beab59379d4f44009498 887 Oct 31 00:12 staging_1lbgjqpp9x_id_rsa
-rw-r--r--. 1 59f7beab59379d4f44009498 59f7beab59379d4f44009498 255 Oct 31 00:12 staging_1lbgjqpp9x_id_rsa.pub
[1lbgky2qks-1lbgky2qks.dms.int.usw2.ficoanalyticcloud.com .ssh]\>
```

Figure 5.12 Listing of the Files with Generated Keys for 3 life-cycle environments

Per DMP Component provider specification, the DMP manager moves all the files under **DMP-INF** (along with the ***config folder, where feConfig.json and subscriptionDetails.json are saved***) during promotion to Staging and Production environments, to the respective component instance gears for staging and production.

Each update to the FEC configuration for higher environments (Staging and Production), requires promotion by the user to upgrade to such environment.

During File Exchange Event operation, the FEC engine under the hood makes use of <code>***DMPContext.getDmpEnivronment( )***</code>to determine the current environment, and makes uses of the environment-specific S3 and NS service instances provisioned for the env-specific FE component gear, to process file transfers.

FEC Retry Mechanism

The underlying design goal for Retry Scheduler is, to be able to efficiently address any network-induced glitches, memory issues on the destination host, and other hard-ware configuration changes (like switches used as part of scaling mechanism in the load-balancer that the destination host is a member of, etc.,), causing File Transfer failures. The time-window allowed for the retry mechanism should be optimal enough to ensure an acceptable trade-off between the production support and FEC product team, is guaranteed, while the client SLA’s are at no risk.

A retry scheduler internal to FEC is configured to retry File exchange/transfer (Sometimes, the word “transfer” is loosely used to mean “exchange” operations, for all the file exchange events, that have STATUS_CODE corresponding to FAILURE or ERROR. A light-weight open-source job scheduler, like, Guava Retrying (an extension to Google's Guava library, with configurable retrying strategies) can be considered.

Alternatively, proven open-source but relatively heavy-weight framework like Quartz Scheduler can also be considered. Both are usable with annotation-based support within widely adopted frameworks like Spring etc.

Considering Guava Retrying, as the retry scheduler adopted, the typical workflow of the retry scheduler is follows.

The input to the Retry Operation is a failed file transfer execution. For the very first time, the file transfer is initiated by the FEC code under the hood, using the config data and file metadata, derived by parsing the JSON payload with notifications for events of type "***s3:ObjectCreated****" on S3, when the notifications are received through the POST rest api invocation by the NS instance servicing the FEC.

On successful file transfer, the file exchange status is set to 'COMPLETED', and on failure of file transfer, the status is initially set to FAILED, and on subsequent failures, status is set to FAILED_RETRY, each time.

After the first failure, FEC retry mechanism kicks off. The retry scheduler will wait for unit of time (in minutes) equal to ***2 to the power of <attempt_no>***, before trying to process file transfer again. In FEC case, ***where, attempt_no = {1, 2, …., k, ….n}, where n = 11, and k >= 1 and k <= n.***

For first retry attempt, it'll be 2 minutes, and for second retry attempt, it'll be 4 minutes and so on and so forth.

As briefly mentioned in section 2.1, FEC makes use of Exponential Backoff-strategy for it's retry operations. 'Retry' mechanism will end, when the kth file transfer attempt is successful, with predefined 'k' no of attempts, or the **MAX_ATTEMPT_NO** is reached, regardless of whether the file transfer has successfully finished or not.

Increasing the max number of retries attempts to 11 (**MAX_ATTEMPT_NO**) should potentially give FEC a time window of 34 hours (i.e., last retry attempt will happen ~34 hours after the first failure). If the retry mechanism should extend beyond this time window, it's worth having a ticket raised by the Production Support folks and address the file transfer failure, by having someone manually transfer the said file/s.

For each failed file transfer attempt, FEC engine throws Java exception and sends an appropriate Java Object with stats of the current transfer attempt, wrapped inside the File exchange runtime exception thus enabling retry mechanism to proceed with it's mechanism, as 'Retry' code will be listening to that.

In case of successful attempts, retry code doesn't throw exception and sends back the Java Object, listening to which Retry will stop, and normal execution resumes.

JSch framework used, will take care of RESUME operation during RETRY event, as discussed in section 5.1.

FEC File Transfer

File exchange necessary involves both inbound and outbound transfer operations, between client machines (external systems) and server machines (machines within the DMP ecosystem).

For NAB, only outbound file transfer is supported. NAB have their own way of doing inbound file transfer, as of today.

For instance, if jsch is used as a tunneling proxy for this, a SOCKS proxy setting on jsch allows the client code to connect to a *running* proxy server on the remote side. An sshd on the remote side would **not** be considered a SOCKS proxy. This is done by establishing a local port forward to the ssh port on the machine that's we're tunneling to, then establish a secondary ssh connection to remove system using the api.

FEC File Transfer Status

When the file transfers have erred out or when the transfer is complete, FE table FE_FILE_EXCH_EVENTS is updated with status of transfer corresponding the EXCH_ID, in terms of STATUS_CODE and RETRY_COUNT. RETRY_COUNT is incremented for each transfer event, in case of failure in file transfer.

When there’s no longer a need to increment RETRY_COUNT, the file transfer has either succeeded, or the MAX_ATTEMPT_NO for retry operation has reached. In the latter case, the file transfer can no longer be attempted, and the final status of the file is marked as FAILED.

Security View

FEC will be secured by DMP's security filter that prevents unauthorized access to provider services.When external clients invoke the target URL, it will include an authorization token, provided by the DMP IAM in the request header for target to authenticate the request.

AWS

It’s the responsibility of the File Exchange component implementation code to ensure, different artifacts taking part in the File Exchange component workflow are made PCI-Compliant and made secure applying DMP standard security policies.

Chapter 6

Openshift k8s Cluster using FICO Web Services

Overview

This chapter contains an overview of the installation guide for deploying kubernetes Openshift clusterfor On-Premises on FICO web services (FWS). This cluster can be further used for deploying DMP.

Create Resources for cluster Deployment

- Login to FWS dashboard and ensure you are in the right project
- Login to the FWS dashboard
- Once logged in make sure you are in the right project by selecting Projects on the right top corner like shown below :

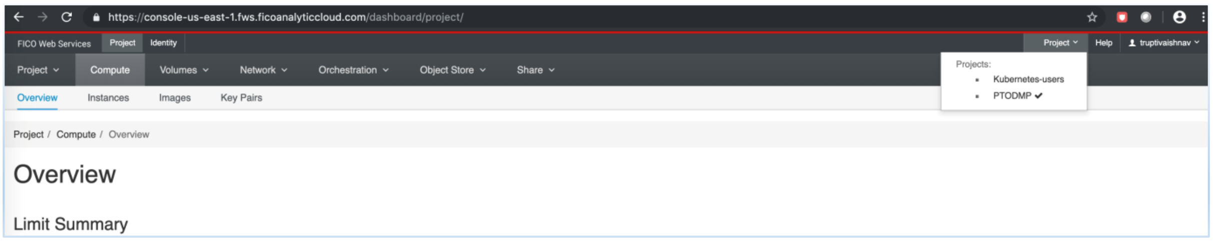


Figure 6.1 FWS Dashboard

Create Master node instances using FWS UI

Based on the requirement, the k8s cluster could have 1 or more master nodes. For the demo purpose we are building a cluster having 2 master nodes with the following configuration:

Master Node (at least 2, recommended 3)	8-vCPU 32 GB RAM 40 GB HDD 400 GB Docker Storage (optional) RHEL 7.5

Table 4 Master Instance Configuration

For creating the master node instances, follow the steps below:

Go to Compute->Instances and then click on “Launch Instance” button

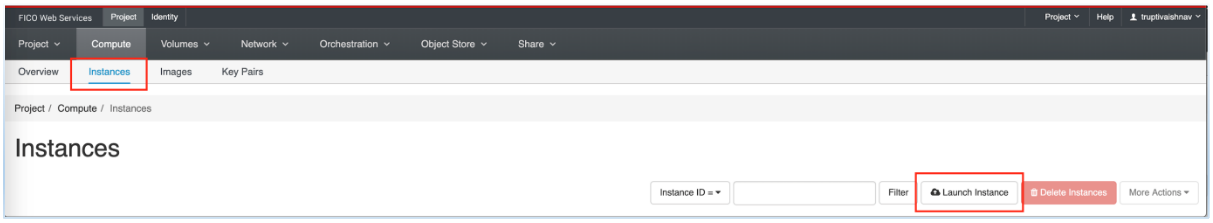
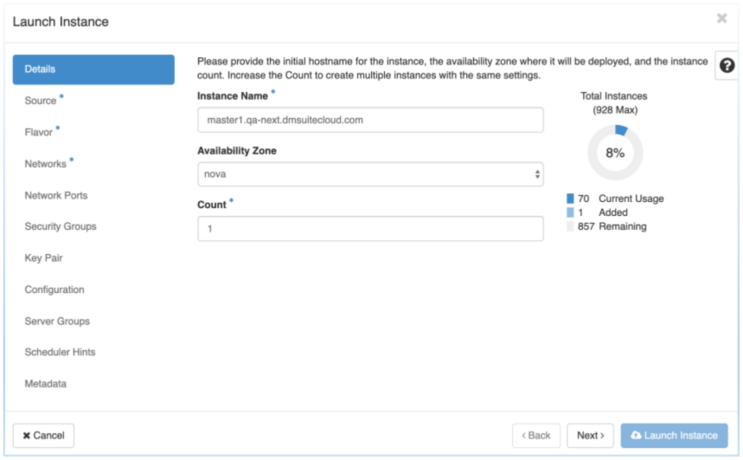


Figure 6.2 Instance Creation Snapshot

1. Launch Instance Wizard – Screen 1 (Details)
   - Enter an ***Instance name*** which is resolvable to corresponding instance IP address. For verification execute the command **$ echo HOSTNAME** on host machine. The output of the command and DNS entry should exactly match for all the nodes.
   - Select ***Availability Zone*** as “nova”
   - Enter ***Count*** as 1. Names of instances should be valid DNS. While creating the instances, don’t use count/replication as this will append the replication count after the hostname resulting in invalid domain name.
   - Click “***Next***” to proceed



Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.

Instance Name * master1.qa-next.dmsuitecloud.com

Availability Zone nova

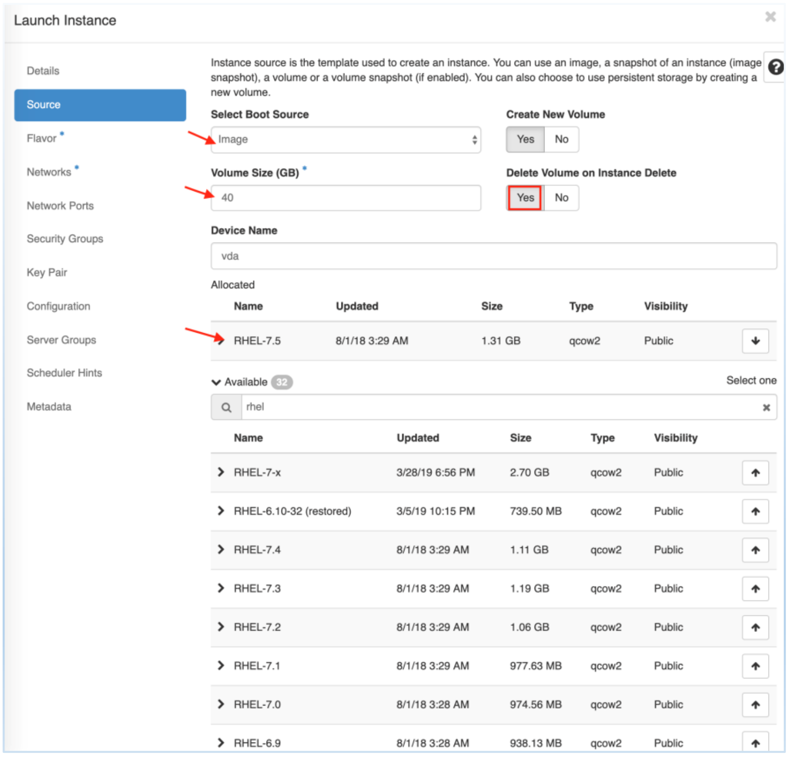
Count * 1

■	70	Current Usage
■	1	Added
□	857	Remaining

Figure 6.3 Instance Creation(Step-1)

2. Launch Instance Wizard – Screen 2 (Source)
   - Select “***Boot Source***” as “***Image***”
   - Enter “***Volume Size***” as “***40”***
   - Select “***Delete Volume on Instance Delete***” as “***Yes***”

- Choose “***RHEL-7.5***” from the list of images
- Click on “***Next***” to proceed.



Launch Instance

Details
Source
Flavor *
Networks *
Network Ports
Security Groups
Key Pair
Configuration
Server Groups
Scheduler Hints
Metadata

Instance source is the template used to create an instance. You can use an image, a snapshot of an instance (image snapshot), a volume or a volume snapshot (if enabled). You can also choose to use persistent storage by creating a new volume.

Select Boot Source: Image

Create New Volume: [✓ Yes] No

Volume Size (GB) *: 40

Delete Volume on Instance Delete: [✓ Yes] No

Device Name: vda

Allocated

Name	Updated	Size	Type	Visibility
RHEL-7.5	8/1/18 3:29 AM	1.31 GB	qcow2	Public

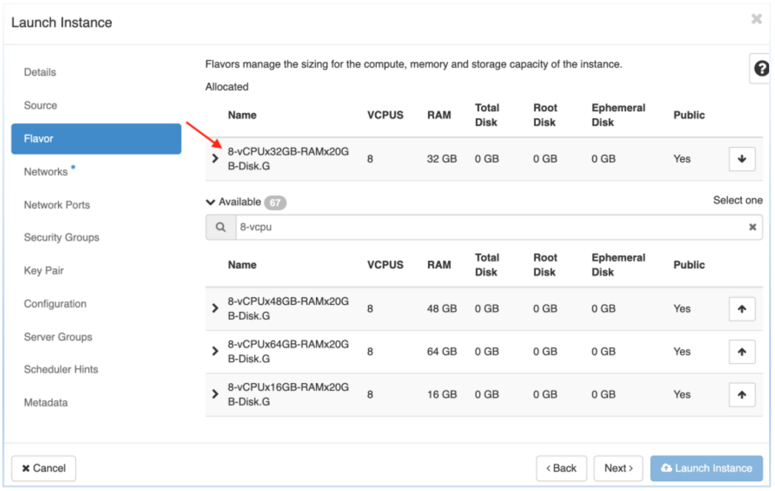
Available 32 — Select one

rhel

Name	Updated	Size	Type	Visibility
RHEL-7-x	3/28/19 6:56 PM	2.70 GB	qcow2	Public
RHEL-6.10-32 (restored)	3/5/19 10:15 PM	739.50 MB	qcow2	Public
RHEL-7.4	8/1/18 3:29 AM	1.11 GB	qcow2	Public
RHEL-7.3	8/1/18 3:29 AM	1.19 GB	qcow2	Public
RHEL-7.2	8/1/18 3:29 AM	1.06 GB	qcow2	Public
RHEL-7.1	8/1/18 3:29 AM	977.63 MB	qcow2	Public
RHEL-7.0	8/1/18 3:28 AM	974.56 MB	qcow2	Public
RHEL-6.9	8/1/18 3:28 AM	938.13 MB	qcow2	Public

Figure 6.4 Instance Creation(Step-II)

3. Launch Instance Wizard – Screen 3 (Flavor)
   - Filter & Select “***8-vCPUx32GB-RAMx20GB-Disk.G***” from the available list
   - Click “***Next***” to proceed



Launch Instance

Details
Source
Flavor
Networks *
Network Ports
Security Groups
Key Pair
Configuration
Server Groups
Scheduler Hints
Metadata

Flavors manage the sizing for the compute, memory and storage capacity of the instance.

Allocated

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
8-vCPUx32GB-RAMx20GB-Disk.G	8	32 GB	0 GB	0 GB	0 GB	Yes

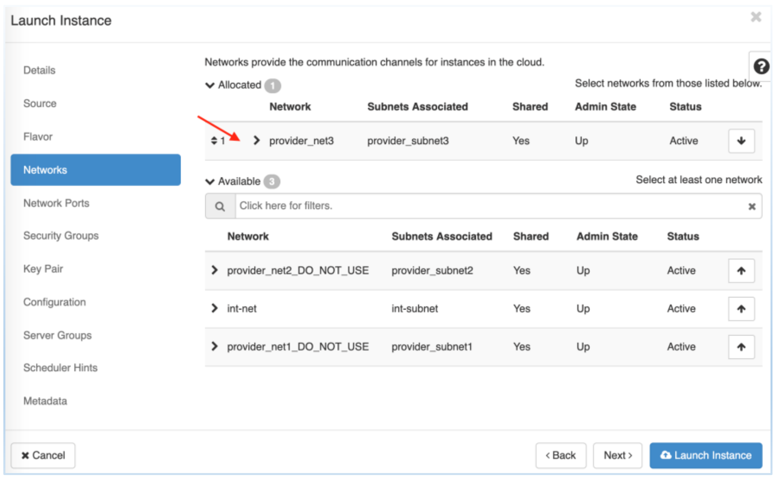
Available 67 — Select one

8-vcpu

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
8-vCPUx48GB-RAMx20GB-Disk.G	8	48 GB	0 GB	0 GB	0 GB	Yes
8-vCPUx64GB-RAMx20GB-Disk.G	8	64 GB	0 GB	0 GB	0 GB	Yes
8-vCPUx16GB-RAMx20GB-Disk.G	8	16 GB	0 GB	0 GB	0 GB	Yes

Figure 6.5 Instance Creation(Step-III)

4. Launch Instance Wizard – Screen 4 (Networks)
   - Filter & Select “***provider_net3***” as the network
   - Click “***Next***” to proceed



Launch Instance

Details
Source
Flavor
Networks
Network Ports
Security Groups
Key Pair
Configuration
Server Groups
Scheduler Hints
Metadata

Networks provide the communication channels for instances in the cloud.

Allocated 1 — Select networks from those listed below.

	Network	Subnets Associated	Shared	Admin State	Status
1	provider_net3	provider_subnet3	Yes	Up	Active

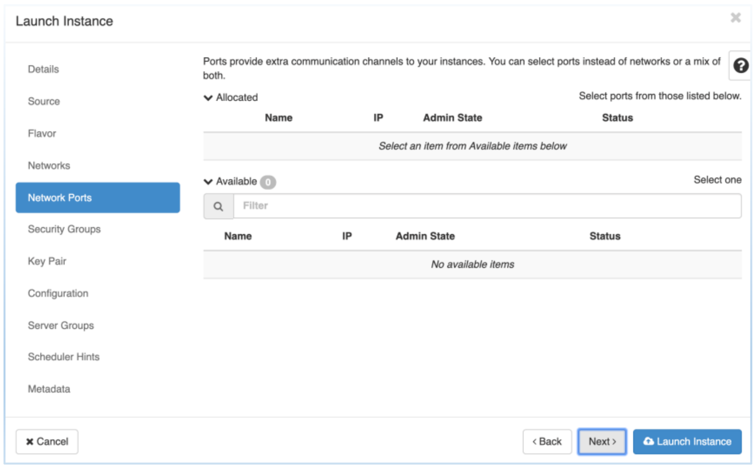
Available 3 — Select at least one network

Click here for filters.

Network	Subnets Associated	Shared	Admin State	Status
provider_net2_DO_NOT_USE	provider_subnet2	Yes	Up	Active
int-net	int-subnet	Yes	Up	Active
provider_net1_DO_NOT_USE	provider_subnet1	Yes	Up	Active

Figure 6.6 Instance Creation(Step-IV)

5. Launch Instance Wizard – Screen 5 (Network Ports)
   - Click “Next” to proceed on this screen.



Launch Instance

Details
Source
Flavor
Networks
Network Ports
Security Groups
Key Pair
Configuration
Server Groups
Scheduler Hints
Metadata

Ports provide extra communication channels to your instances. You can select ports instead of networks or a mix of both.

Allocated — Select ports from those listed below.

Name	IP	Admin State	Status
Select an item from Available items below			

Available 0 — Select one

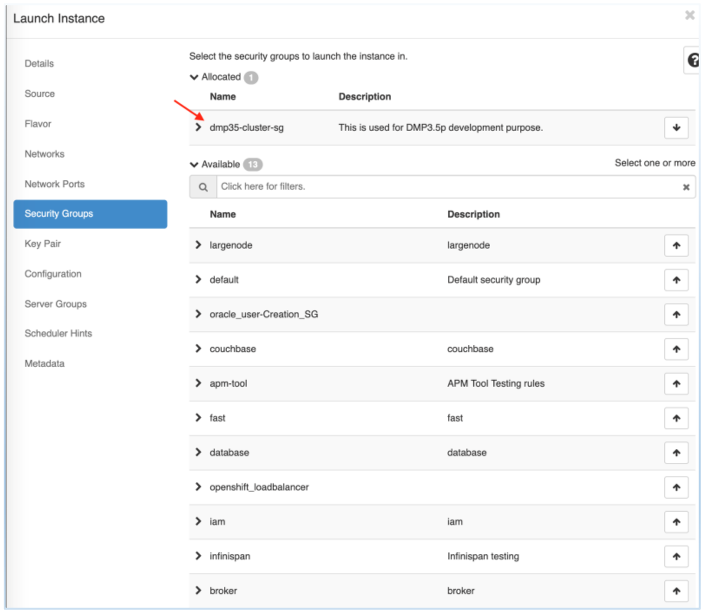
Filter

Name	IP	Admin State	Status
No available items			

Figure 6.7 Instance Creation(Step-V)

6. Launch Instance Wizard – Screen 4 (Security Groups)
   - While creating the instances, create new security group if required else leave it as default. For cluster installation you must create a Security group with port 8443 as open port (Link). For testing purpose use ***dmp35-cluster-sg***.

- Click “**Next**” to proceed



Launch Instance

Select the security groups to launch the instance in.

Allocated 1

Name	Description
dmp35-cluster-sg	This is used for DMP3.5p development purpose.

Available 13 — Select one or more

Click here for filters.

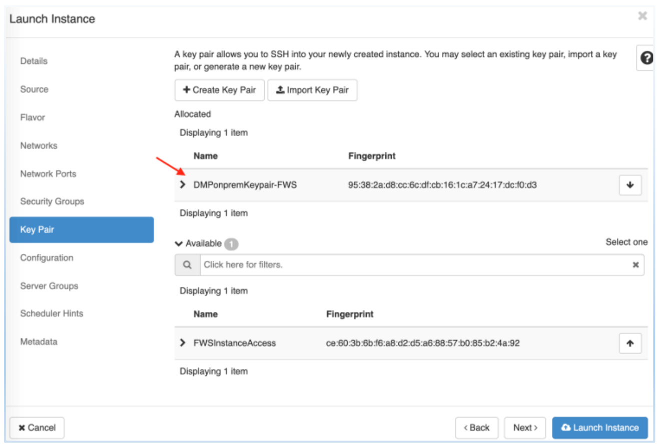
Name	Description
largenode	largenode
default	Default security group
oracle_user-Creation_SG	
couchbase	couchbase
apm-tool	APM Tool Testing rules
fast	fast
database	database
openshift_loadbalancer	
iam	iam
infinispan	Infinispan testing
broker	broker

Figure 6.8 Instance Creation(Step-VI)

7. Launch Instance Wizard – Screen 6 (Key Pair)

   Either create a new key pair or import existing key pair for ssh-ing into your instances.

   - To create a new key use “Create Key Pair” button & create one.
   - Filter & Select the newly created keypair from the available options.
   - If you want to use the existing keypair then use “Import Key Pair” button on the screen.
   - Click “Next” to proceed.


Launch Instance

A key pair allows you to SSH into your newly created instance. You may select an existing key pair, import a key pair, or generate a new key pair.

Allocated

Displaying 1 item

Name	Fingerprint
DMPonpremKeypair-FWS	95:38:2a:d8:cc:6c:df:cb:16:1c:a7:24:17:dc:f0:d3

Displaying 1 item

Available 1 — Select one

Click here for filters.

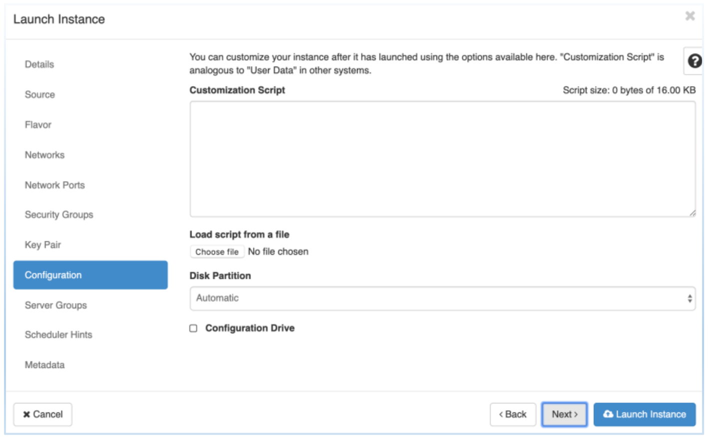
Displaying 1 item

Name	Fingerprint
FWSInstanceAccess	ce:60:3b:6b:f6:a8:d2:d5:a6:88:57:b0:85:b2:4a:92

Displaying 1 item

Figure 6.9 Instance Creation(Step-VII)

8. Launch Instance Wizard – Screen 8 (Configuration)
   - Click “Next” to proceed on this screen.


Launch Instance

You can customize your instance after it has launched using the options available here. "Customization Script" is analogous to "User Data" in other systems.

Customization Script — Script size: 0 bytes of 16.00 KB

Load script from a file

Choose file — No file chosen

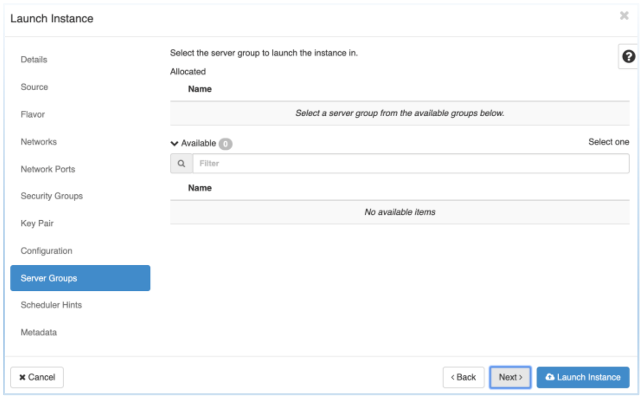
Disk Partition

Automatic

☐ **Configuration Drive**

Figure 6.10 Instance Creation(Step-VIII)

9. Launch Instance Wizard – Screen 9 (Server Groups)
   - Click “Next” to proceed on this screen.


Select the server group to launch the instance in.

Allocated

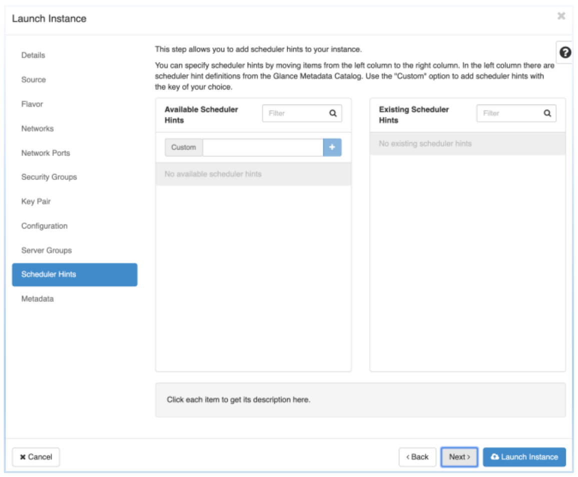
Select a server group from the available groups below.

Available 0

No available items

Figure 6.11 Instance Creation(Step-IX)

10. Launch Instance Wizard – Screen 10 (Scheduler hints)
    - Click “Launch Instance” to proceed on this screen.


This step allows you to add scheduler hints to your instance.

You can specify scheduler hints by moving items from the left column to the right column. In the left column there are scheduler hint definitions from the Glance Metadata Catalog. Use the "Custom" option to add scheduler hints with the key of your choice.

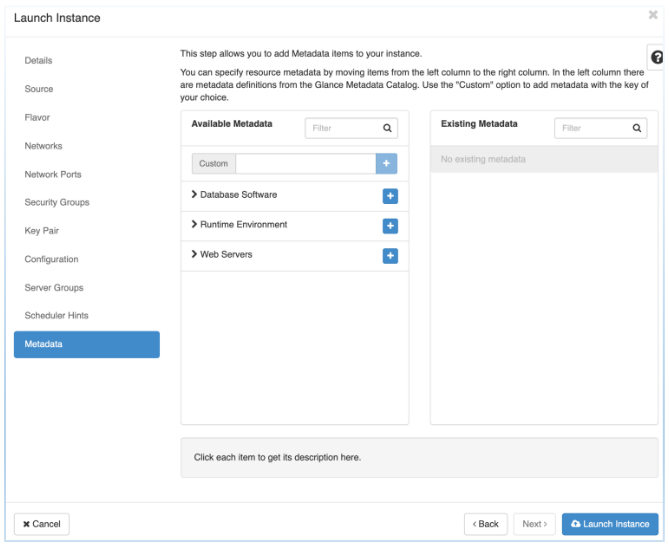
No available scheduler hints

No existing scheduler hints

Click each item to get its description here.

Figure 6.12 Instance Creation(Step-X)

11. Launch Instance Wizard – Screen 11 (Metadata)
    - Click “Next” to proceed on this screen.



This step allows you to add Metadata items to your instance.

You can specify resource metadata by moving items from the left column to the right column. In the left column there are metadata definitions from the Glance Metadata Catalog. Use the "Custom" option to add metadata with the key of your choice.

No existing metadata

Click each item to get its description here.

Figure 6.13 Instance Creation(Step-XI)

Repeat steps 1)-12) for creating another instance of master node.

Create Infrastructure Node instances using FWS UI

We would be creating 2 infrastructure nodes in this cluster with the following configurations:

Infrastructure Node **(at least 2, recommended 3)**	**8-vCPU** **32 GB RAM** **40 GB HDD** **400 GB Docker Storage (optional)** **RHEL 7.5**

Table 5

Repeat steps 1)-12) for creating each instance of infrastructure node.

Create Application Node instances using FWS UI

We would be creating 6 application nodes in this cluster with the following configurations:

Application Node **(at least 6)**	**8-vCPU** **32 GB RAM** **40 GB HDD** **400 GB Docker Storage (optional)**

	RHEL 7.5

Table 6

Repeat steps 1)-12) for creating each instance of application node.

Create Load balancer Node instances using FWS UI

We would be creating 2 load balancer nodes (internal & external) in this cluster with the following configurations:

Load-balancer Nodes **Internal and External** **(one each)**	**4-vCPU** **8 GB RAM** **40 GB HDD** **RHEL 7.5**

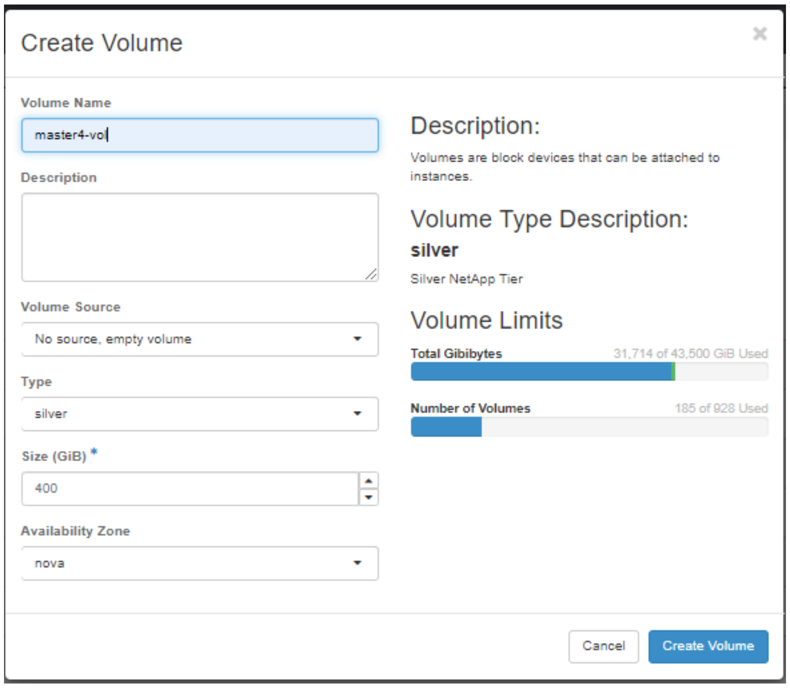
Table 7

Repeat steps 1)-12) for creating each instance of load balancer node

Create and Attach Volume to instances

Create volumes of required size (For example: 400 GB) for storing docker images **for master, application** and **infrastructure nodes** and attach it to the respective instances.

For example: Parameters to create volume for master node would be:



Create Volume

Volume Name: master4-vol

Description:

Volume Source: No source, empty volume

Type: silver

Size (GiB) *: 400

Availability Zone: nova

Description:

Volumes are block devices that can be attached to instances.

Volume Type Description:

silver

Silver NetApp Tier

Volume Limits

Total Gibibytes — 31,714 of 43,500 GiB Used

Number of Volumes — 185 of 928 Used

Figure 6.14 Volume Creation(Step-I)

Create NFS Share for the cluster

Click on Share-Shares tab and then user "Create Share" button to create the NFS share

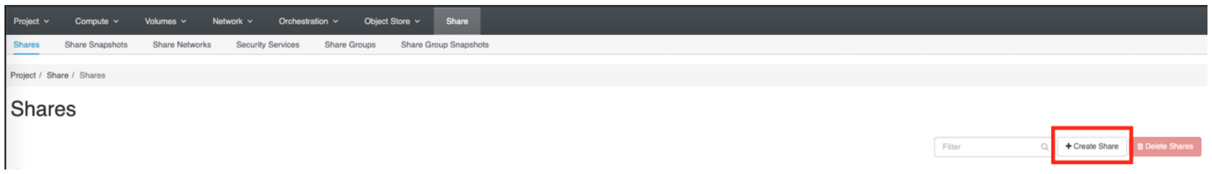
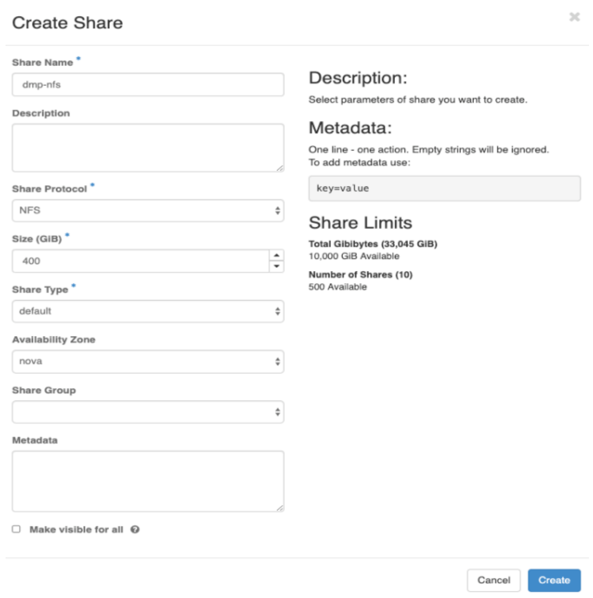


Figure 6.15 NFS Creation(Step-I)

- Enter the "***Share Name***"
- Select "***Share Protocol***" as "***NFS***"
- Enter "***Size***" as "***400***"
- Select "***Availability Zone***" as "***nova***"



Create Share

Share Name *
dmp-nfs

Description

Share Protocol *
NFS

Size (GiB) *
400

Share Type *
default

Availability Zone
nova

Share Group

Metadata

☐ Make visible for all

Description:
Select parameters of share you want to create.

Metadata:
One line - one action. Empty strings will be ignored.
To add metadata use:

key=value

Share Limits
Total Gibibytes (33,045 GiB)
10,000 GiB Available
Number of Shares (10)
500 Available

Cancel
Create

Figure 6.16 NFS Creation(Step-II)

DNS Entry for all instances in Route53

Add DNS entry in Route53 for all of the instances, resolving to instance IP address. Follow this AWS guide for detailed steps:
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-creating.html

Remove any wildcard DNS if present from Route53 before running Ansible playbook. For example remove this record set: *.<deployment-zone>.

Note: Sign in to the DMSCloud AWS Management Console and open the Route 53 console at https://dmscloud.signin.aws.amazon.com/console

Prerequisites

Install Ansible

Ansible is easy open supply IT engine that automates application preparation, intra service orchestration, cloud provisioning and lots of alternative IT tools

- brew install ansible (Mac)
- sudo easy_install pip | sudo pip install ansible (others)
- For Ansible installation in Windows 10, follow this link: https://www.jeffgeerling.com/blog/2017/using-ansible-through-windows-10s-subsystem-linux

Install certbot

Certbot is EFF's tool to get certs from Let's write in code and (optionally) auto-enable HTTPS on your server.

- Install certbot for creating certificates using following commands:
- brew install certbot
- pip install certbot

Clone the on-prem installer repository

- git clone <url>
- git fetch all
- git checkout <branch-name>
- All the installation scripts for cluster setup are located in "**playbook"** directory.

Configuration of the Installer

- Add keys to the installer
- Fetch the private and public keys from FWS and place them in the cloned repo
- Configure inputs required by installer

Update the **roles/common/vars/main.yml** with the inputs required by the installer

INPUTS	Description
DomainName	Parent-domain name
RHEL_username	Red Hat Enterprise Linux email address

RHEL_password	Red Hat Enterprise Linux password
docker_storage	1. Internal- When no additional volume used. 2. External - When additional volume used
docker_volume	Mount point of docker volumne Eg. /dev/vdb
OPENSHIFT_MASTER_CLUSTER_PUBLIC_HOSTNAME	External load balancer hostname
Haproxy	enabled – when using haproxy disabled – when not using haproxy
cluster_admin_user_name	Openshift Dashboard login user_name
cluster_admin_user_password	Openshift Dashboard login password
oc_insecureSkipTlsVerify	true always
pvServerIP	IP address of PV server. Use 10.106.32.5 or 10.106.32.6 IP address to mount the shares.
defaultDirectory	Directory in PV Server. This can be fetched from Project – Shares – Share Details: NFS Share. defaultDirectory value will be the path after the IP mentioned above.

Table 7

- Add certificates in the installer.
- Certs directory (Absolute path: **<path>**) should contain the following:
- cert-bundle.pem
- cert.pem
- key.pem

Execution of the installer

```
ansible-playbook -i hosts main.yml --private-key=keys/id_rsa
```

Note: The command execution will take around 1.5 hours.

Open Openshift dashboard

- After the setup is done open the url: **<externallb-hostname>:8443/console**to open the Openshift dashboard.
- Login using the credentials.

Chapter 7

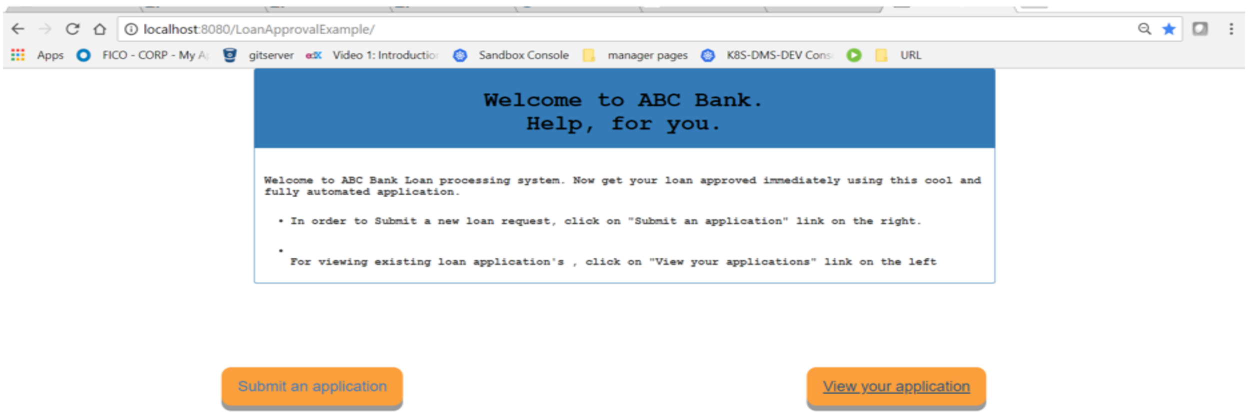
LoanApprovalExample Demo

Problem Statement: ABC Bank needs to get their Loan processing completely revamped. The current process involves lot of human intervention and it takes considerably longer time to process a new loan request. ABC Bank wants to completely automate the process. You have to design and develop an end to end solution based on JEE framework and Score calculation (model development).

UseCases:

UC001: Submit new application

1. User access the application
2. The application displays following links to the user
    a. Submit new application
    b. View existing applications
3. User clicks Submit new application
4. A form is displayed to the user with information related two name, address, loan amount etc.
5. User fills the data and click on “Submit” button
6. Once the application is submitted, user gets a message that application is submitted successfully.



Welcome to ABC Bank.
Help, for you.

Welcome to ABC Bank Loan processing system. Now get your loan approved immediately using this cool and fully automated application.

- In order to Submit a new loan request, click on "Submit an application" link on the right.
- For viewing existing loan application's , click on "View your applications" link on the left

Figure 7.1 Loan Application Home Page

UC002: Submit new application - Incomplete data flow

1. User does Step 1 to 5 mentioned in UC001
2. If user doesn't supply the mandatory information, user will be displayed list of error of which fields are missing
3. User corrects the errors and click Submit

4. Once the application is submitted, user gets a message that application is submitted successfully.

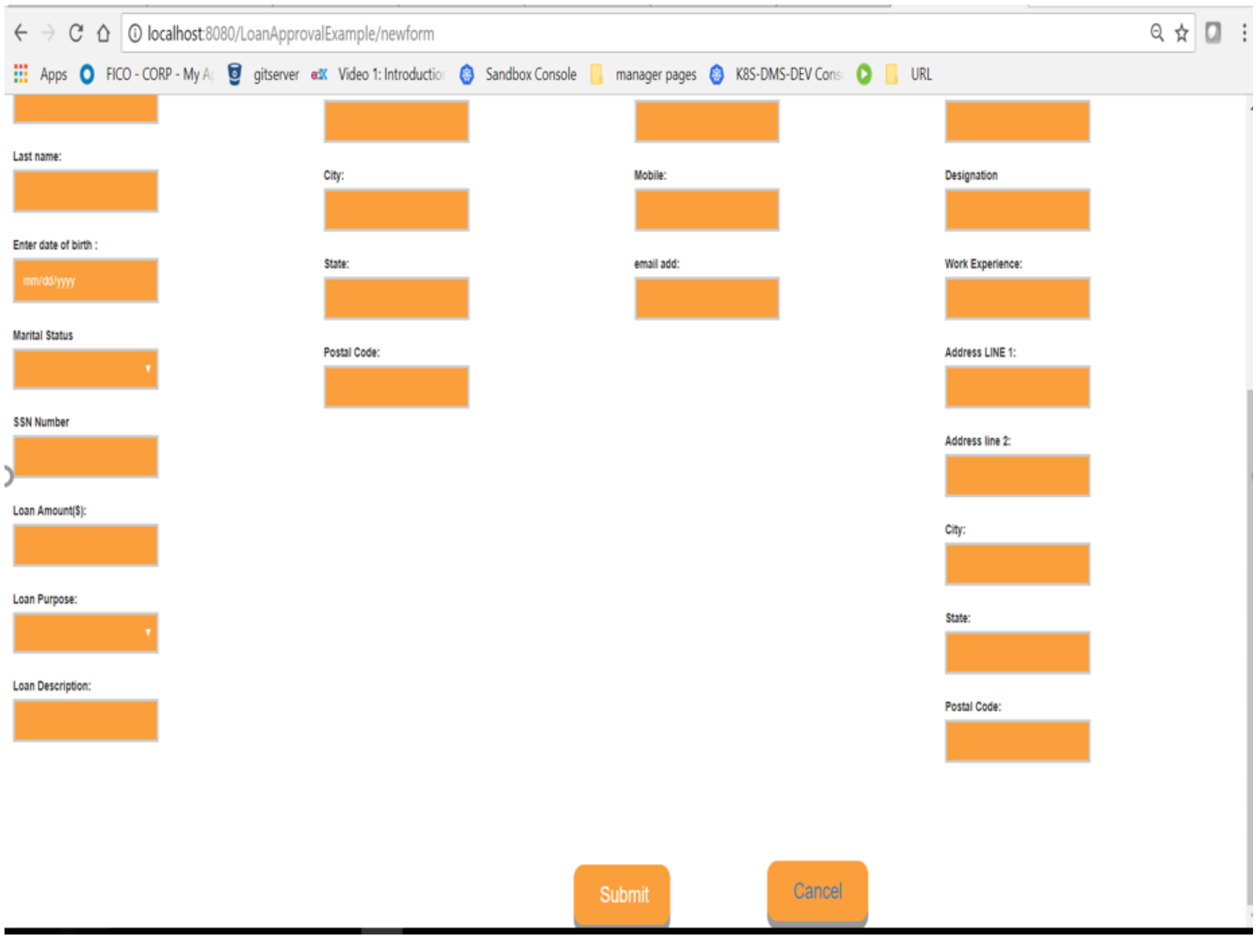


Figure 7.2 Form to Submit Application

UC003: View existing applications

1. User access the application
2. The application displays following links to the user
    a. Submit new application
    b. View existing applications
3. User clicks View existing applications.
4. A grid will be displayed to the user which will have following information

a. Application ID (auto generated by system when application is submitted)
b. Applicant Name – First name + Middle Name + Last Name
c. Application Submitted date (Auto generated)
d. Application Status – (Following status are possible – Processing, Accept and Decline)

UC004: View existing application

Parent Story UC003

1. User performs the operation of UC003
2. User clicks on the Application ID in the grid.
3. User gets the detail of the application in read only format along with Application ID, Score, Status and Decline Reason. If the application is Approved, the decline reason will not have any value.

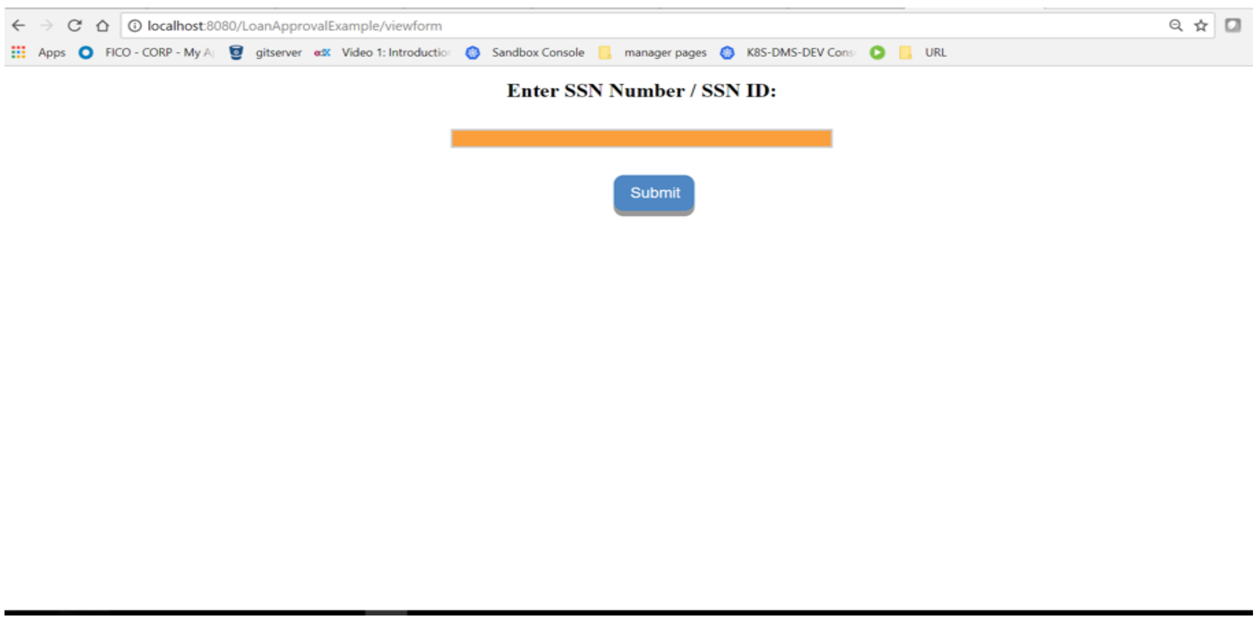


Figure 7.3 View existing application

Score Calculation and Integration

Score calculation will be done on the model (see model Development Flow) that you develop using the data (credit bureau) provided in database/file.

Instruction for Scoring

1. Combining with the bureau data based on the SSN (id)
2. Variable generation (combination of multiple fields), similar to the variable generation part during the modeling phase
3. Computing text fields (Text analytics) using the same process as in the modeling flow.
4. Computing the score using the model trained in Excel

Once the scoring is complete compare with a threshold to indicate Acceptance / Rejection of the application. The threshold should be determined based on the default rate in the modeling data.

Application Architecture

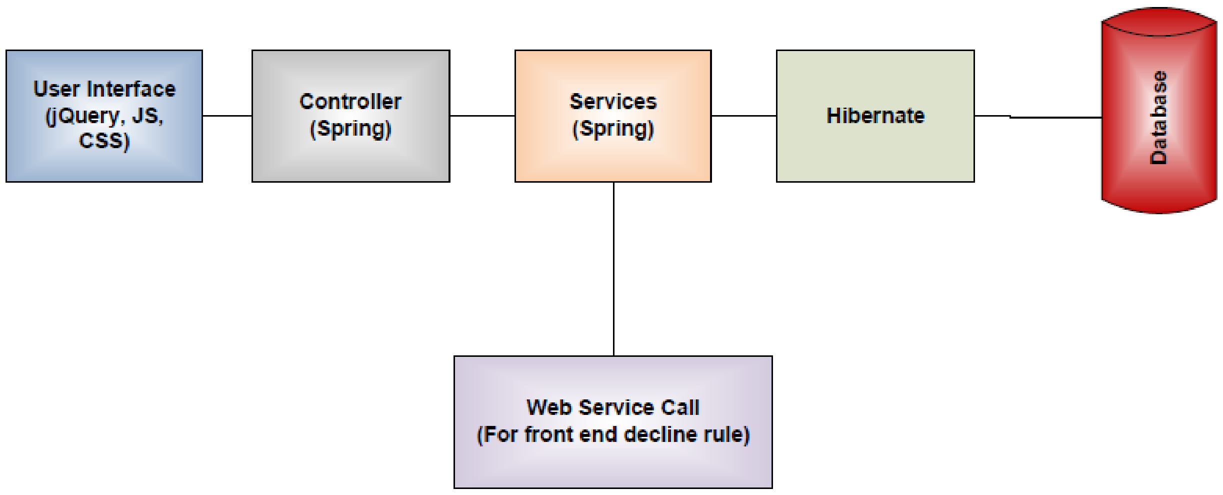

User Interface (jQuery, JS, CSS)
Controller (Spring)
Services (Spring)
Hibernate
Database
Web Service Call (For front end decline rule)

Figure 7.4 Application Architecture

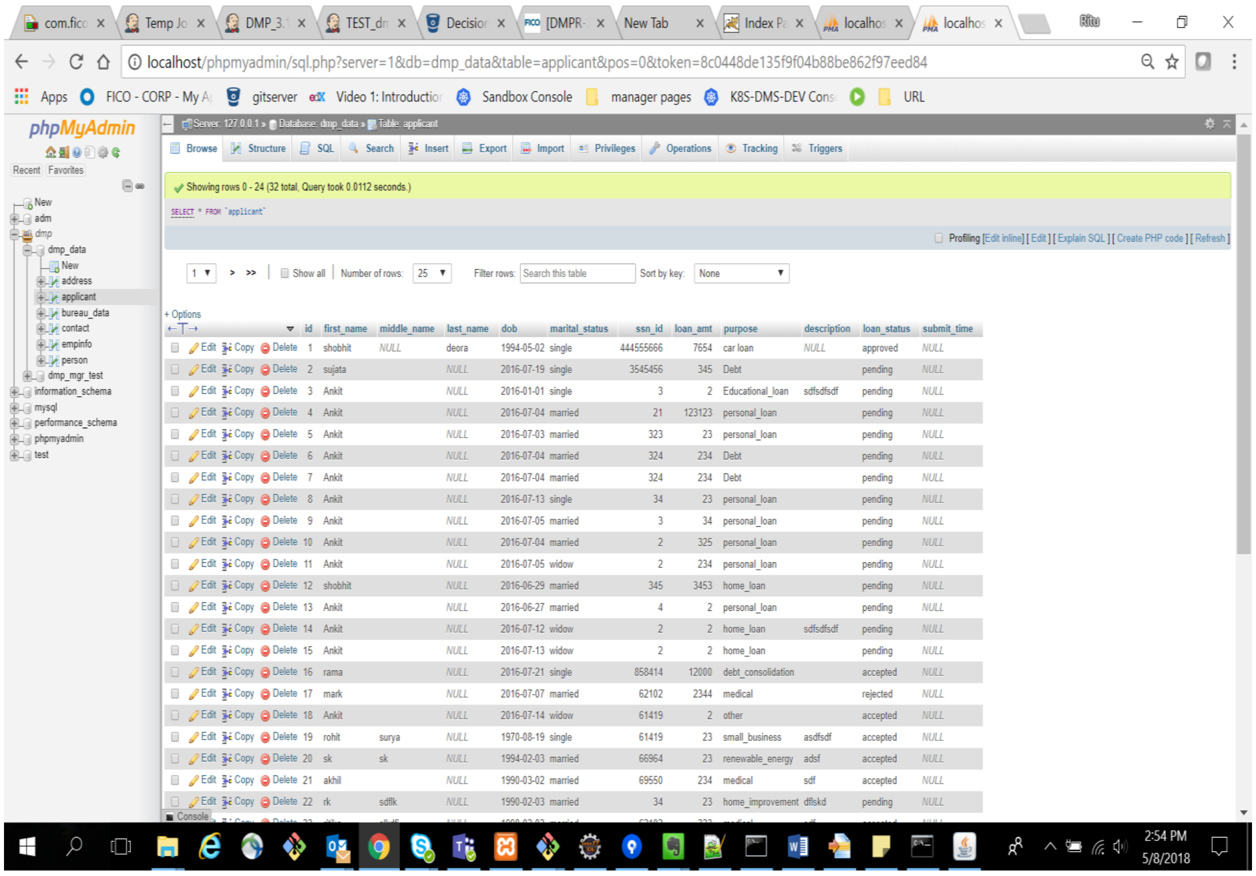
Technologies Used:

- User Interface – JSP, CSS, Java scripts
- Backend – Spring + Hibernate
- Database – Oracle
- Front end decline rule invocation – Web Services and XML parsers
- Scoring Models – Java
- Modeling - Excel

Front End Decline Rules The application should be declined if

- Applicant's age is <18 or >65.
- Applicant's work experience is < 6 months
- Applicants annual salary is < $10,000

Database Attached to the Application:



localhost/phpmyadmin/sql.php?server=1&db=dmp_data&table=applicant&pos=0&token=8c0448de135f9f04b88be862f97eed84

Server: 127.0.0.1 » Database: dmp_data » Table: applicant

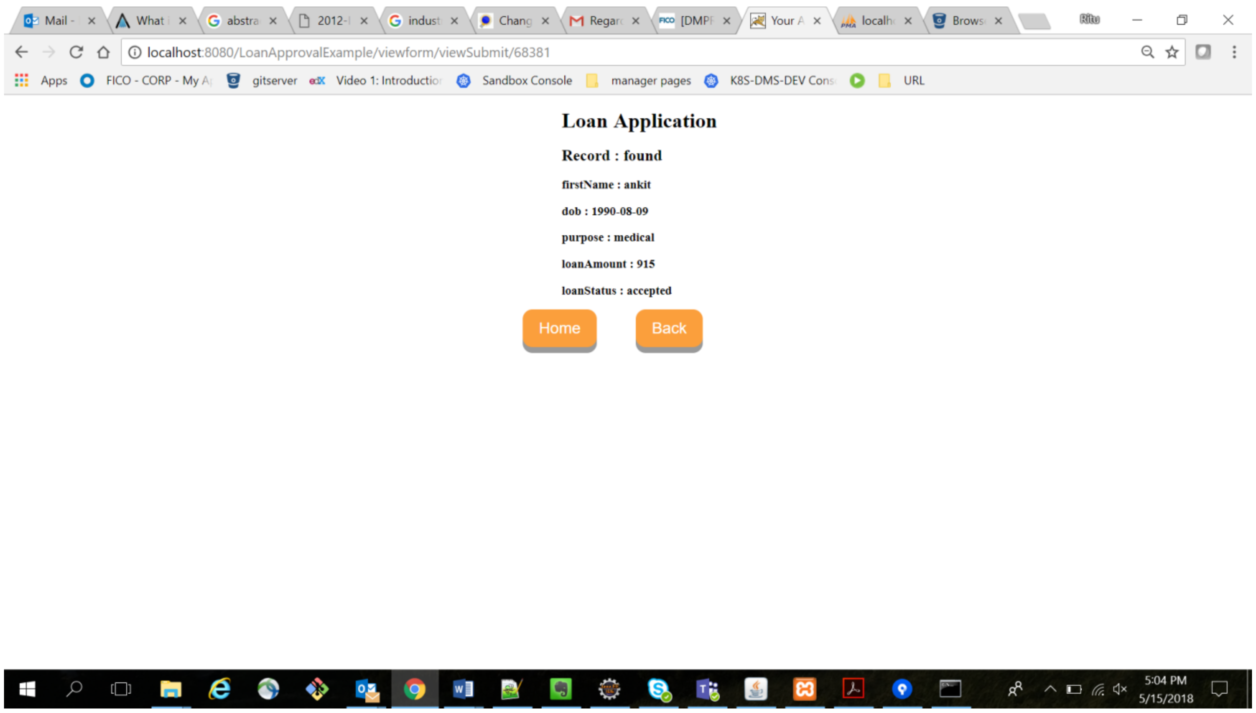
Showing rows 0 - 24 (32 total, Query took 0.0112 seconds.)

```
SELECT * FROM `applicant`
```

Number of rows: 25 Filter rows: Search this table Sort by key: None

id	first_name	middle_name	last_name	dob	marital_status	ssn_id	loan_amt	purpose	description	loan_status	submit_time
1	shobhit	NULL	deora	1994-05-02	single	444555666	7654	car loan	NULL	approved	NULL
2	sujata		NULL	2016-07-19	single	3545456	345	Debt		pending	NULL
3	Ankit		NULL	2016-01-01	single	3	2	Educational_loan	sdfsdfsdf	pending	NULL
4	Ankit		NULL	2016-07-04	married	21	123123	personal_loan		pending	NULL
5	Ankit		NULL	2016-07-03	married	323	23	personal_loan		pending	NULL
6	Ankit		NULL	2016-07-04	married	324	234	Debt		pending	NULL
7	Ankit		NULL	2016-07-04	married	324	234	Debt		pending	NULL
8	Ankit		NULL	2016-07-13	single	34	23	personal_loan		pending	NULL
9	Ankit		NULL	2016-07-05	married	3	34	personal_loan		pending	NULL
10	Ankit		NULL	2016-07-04	married	2	325	personal_loan		pending	NULL
11	Ankit		NULL	2016-07-05	widow	2	234	personal_loan		pending	NULL
12	shobhit		NULL	2016-06-29	married	345	3453	home_loan		pending	NULL
13	Ankit		NULL	2016-06-27	married	4	2	personal_loan		pending	NULL
14	Ankit		NULL	2016-07-12	widow	2	2	home_loan	sdfsdfsdf	pending	NULL
15	Ankit		NULL	2016-07-13	widow	2	2	home_loan		pending	NULL
16	rama		NULL	2016-07-21	single	858414	12000	debt_consolidation		accepted	NULL
17	mark		NULL	2016-07-07	married	62102	2344	medical		rejected	NULL
18	Ankit		NULL	2016-07-14	widow	61419	2	other		accepted	NULL
19	rohit	surya	NULL	1970-08-19	single	61419	23	small_business	asdfsdf	accepted	NULL
20	sk	sk	NULL	1994-02-03	married	66964	23	renewable_energy	adsf	accepted	NULL
21	akhil		NULL	1990-03-02	married	69550	234	medical	sdf	accepted	NULL
22	rk	sdflk	NULL	1990-02-03	married	34	23	home_improvement	dflskd	pending	NULL

Figure 7.5 Application Database


localhost:8080/LoanApprovalExample/viewform/viewSubmit/68381

Loan Application

Record : found

firstName : ankit

dob : 1990-08-09

purpose : medical

loanAmount : 915

loanStatus : accepted

Figure 7.6 Loan application in pending state

Steps for Kubernetes deployment of Loan Approval Applications using the following steps:

1. Build a WAR file of the Loan Approval application
2. Build the docker image of the application
3. Create a kubernetes cluster (no need if your minishift is already running)
4. Create deployment for the app
5. Create service for the app
6. Create route for the app
7. Resolve route DNS
8. Connect it with the MySQL database

Chapter 7

CONCLUSION

In these four months I have learned not only technical details but also the corporate policies and formalities. It helped me to understand the responsibilities of several designated persons, as well as mine as an intern. In this phase I was asked to concentrate on my learning. I learnt Kubernetes container-orchestration system, openshift, Docker, Jenkins, ansible, helm, Java and spring framework during this internship. Hopefully, with this organization I could get a better opportunity to learn and grow.

REFERENCES

[1] https://www.wikihow.com/Run-Regression-Analysis-in-Microsoft-Excel

[2] https://dzone.com/articles/a-guide-to-mocking-with-mockito

[3] https://blog.knoldus.com/2017/how-to-integrate-your-maven-project-with-sonarqube/

[4] https://kubernetes.io/docs

[5] https://www.mkyong.com/spring/maven-spring-jdbc-example/

[6] https://www.mkyong.com/webservices/jax-rs/jersey-hello-world-example/