MMITrading Application Deployment on Enterprise Container Solution (OpenShift)

-- Ishant Gaurav

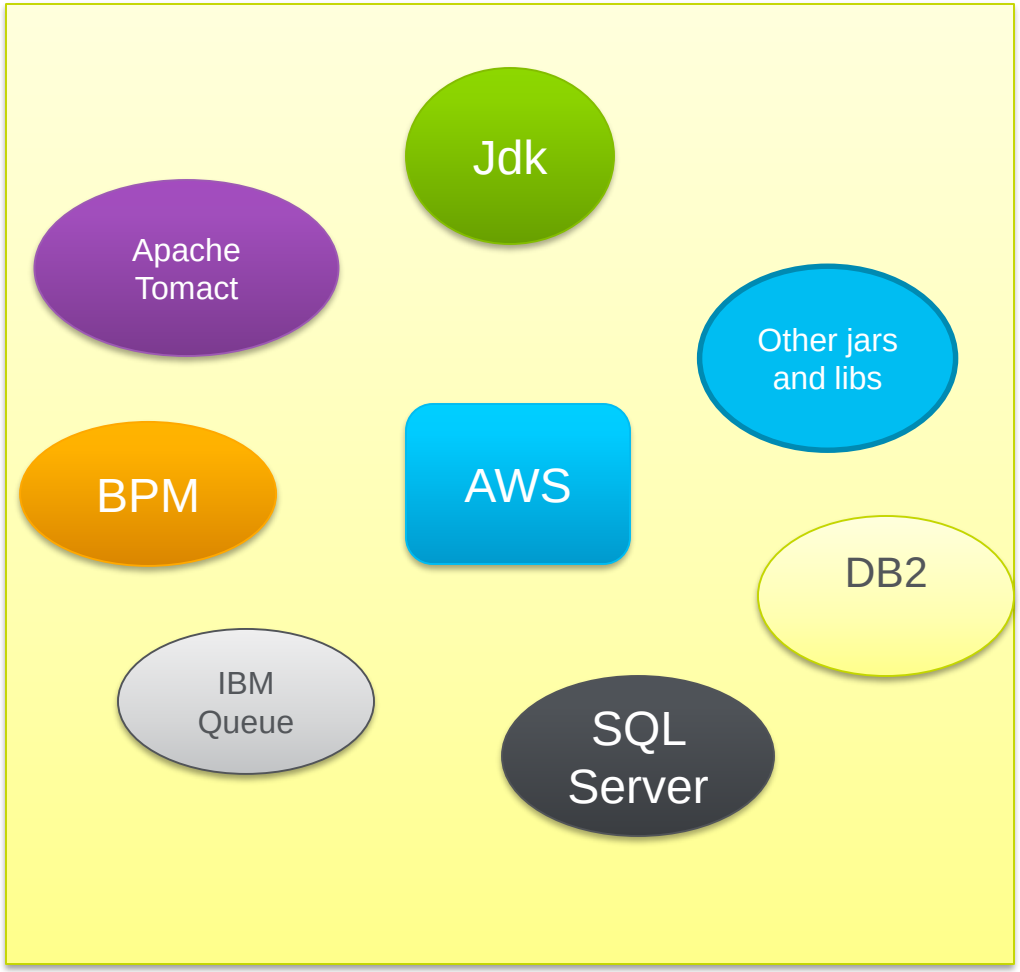
Agenda

- What is the problem we are trying to solve ?
- Why we need to go for Enterprise container solution (OpenShift/Docker)?
- Brief about Docker, Kubernetes and OpenShift
- Single Click Deployment Flow on OpenShift.
- Enterprise Container Solution – CI/CD Architecture.
- MMITrading Application Demo on OpenShift.
- Detailed Steps to deploy application on OpenShift.
- Build Docker Image for OpenShift with ICG Build
- Udeploy Configuration to deploy images
- Q & A session?

The Problem

- IT delivers thousand of application every year to meet the need of business.
- These application require complicated collaborations during installation and integration every time they are deployed .
- To deploy, configure, manage and maintain these complexity takes :
- People
- Expertise
- The right system, infrastructure and architecture.

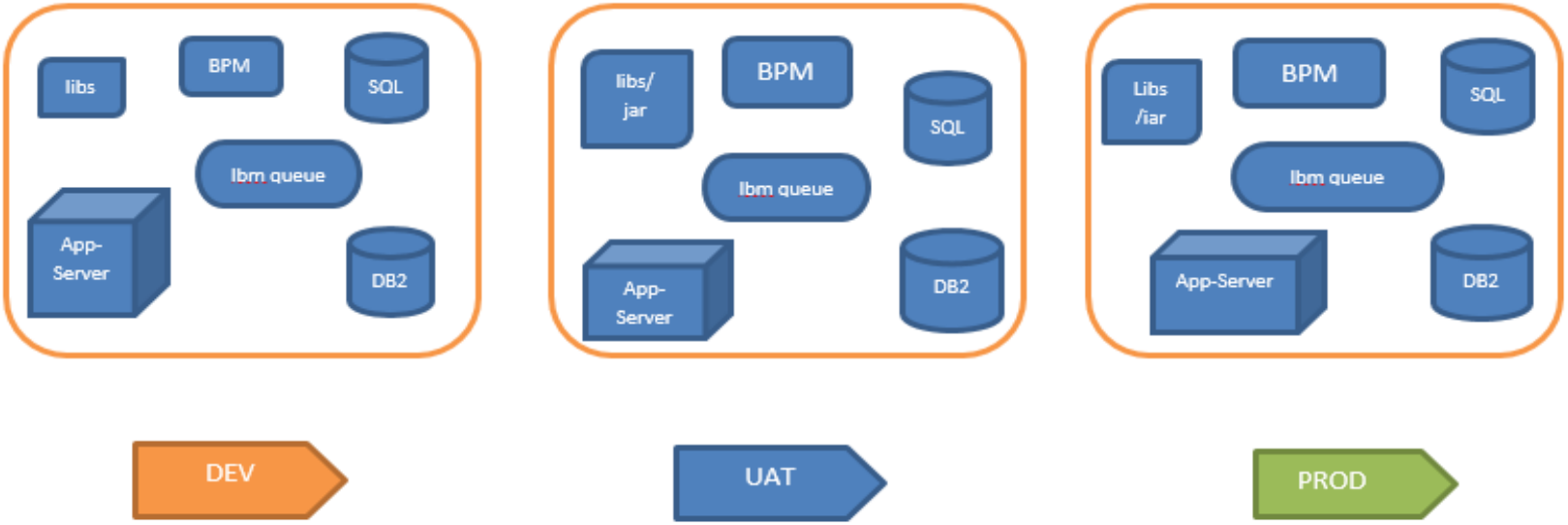
A complicated and costly task.



The Problem

- An enterprise application has to go through different environment like DEV, TEST, UAT before going to production.
- Setting up each environment requires a lot of effort like we have to do the required configuration on each of the environment, all the required libraries has to be maintained for each of them.
- Mostly it has been seen that due to different security cocnern , DEV has different libraries and UAT has different confiuration and libraries which has created a lot of problem for the developers.
- A whole team has to create to take care of the complete end to end build and deployment.
- Still feasible for moniolithic application but becomes very difficult to maintain the same when we go for the microservices architecture.

The Problem



- We would have to configure app-server, database, ibm-queue and BPM for all the environment separately.
- Need to manage the specific external libraries for each environment.
- Sometimes our local and dev environment are on windows whereas UAT and PROD becomes very difficult to manager the software and libs.

Why we need to go for Enterprise Container Solution .

- Containerizing an application makes it very easy to ship and deploy.

Step 1

Package

Installation and configuration of applications is complicated and time consuming, but doing it once is more efficient.

Step 2

Share

Easily share applications between architecture, development, security, and operations teams. Quickly experiment with new applications.

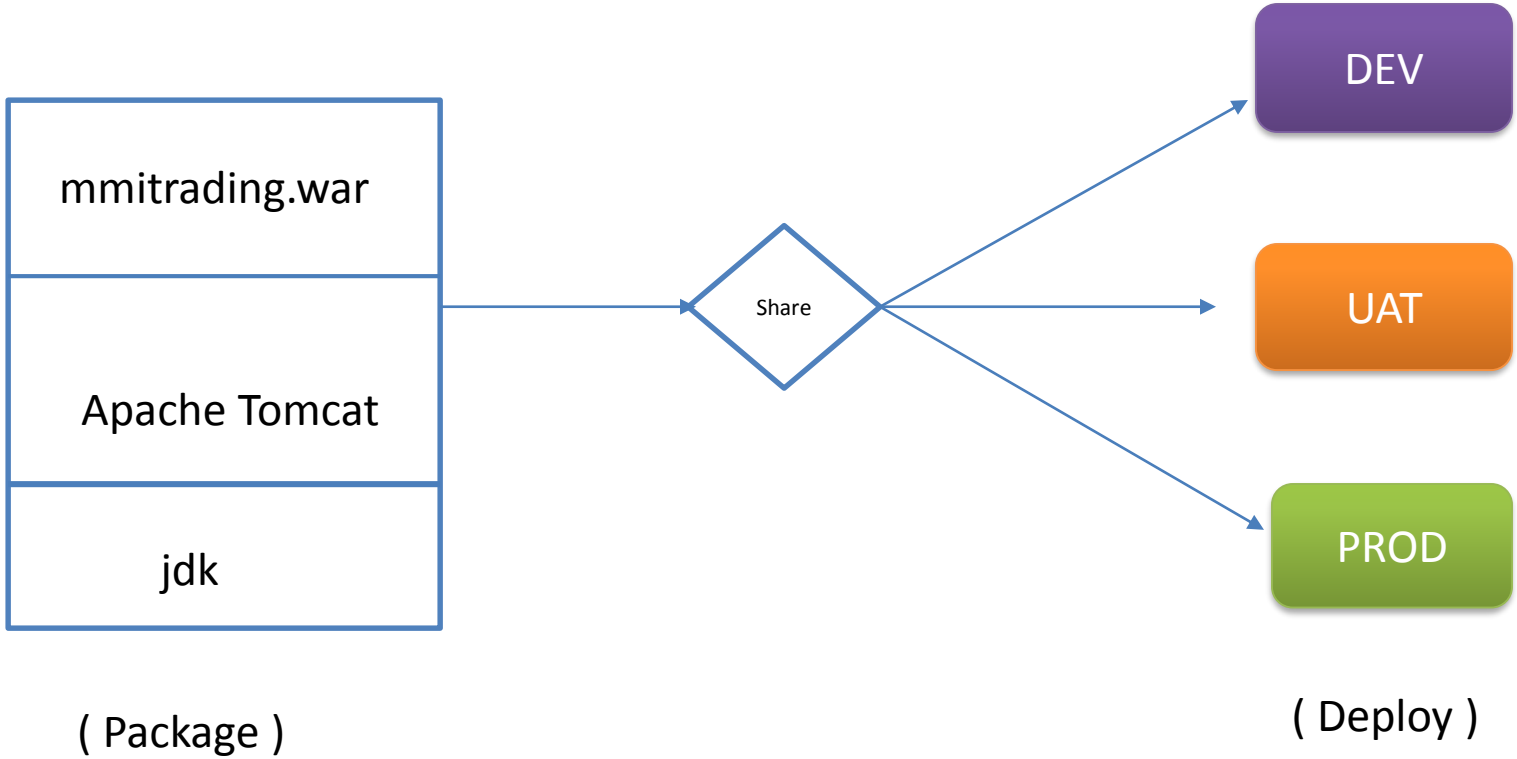
Step 3

Deploy

Deploy new or existing applications in seconds. All of the heavy lifting was done during the build.

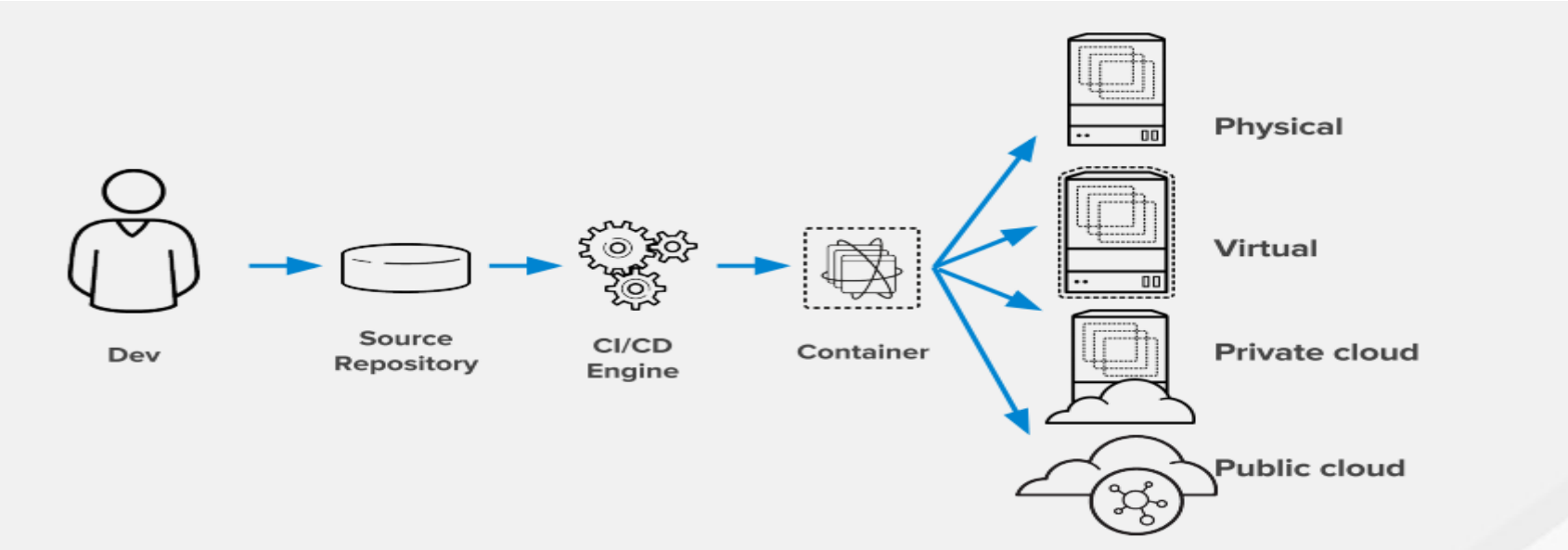
Why we need to go for Enterprise Container Solution .

- We package all the environmental dependency of mmitrading like jdk ,tomcat and war file into one container and then share it among all the environment and simply deploy it.



Why we need to go for Enterprise Container Solution.

- Enable efficiency and automation for microservices but also support traditional applications.
- Enable faster and more consistent deployment.
- Enable application portability across 4 infrastructure footprints: Physical , Virtual, Private and Public cloud.



Brief about Docker, Kubernetes and OpenShift

- **What is Docker** ?
- Docker containers allow virtualization of the OS, allowing each application to be run in a smaller compartment isolated from the other applications
- Docker containers run an image, which is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings.
- **What is Kubernetes ?**
- Kubernetes is an open source platform that automates Linux container operations
- It eliminates many of the manual processes involved in deploying and scaling containerized applications
- It helps with clustering together groups of hosts running Linux containers, and makes the management of this clusters much easier
- **Why do we need Kubernetes ?**
- Real applications apps span across multiple containers
- The containers must deployed across multiple servers
- Kubernetes orchestration allows you to build application services that span multiple containers, schedule those containers across a cluster, scale those containers, and manage the health of those containers over time

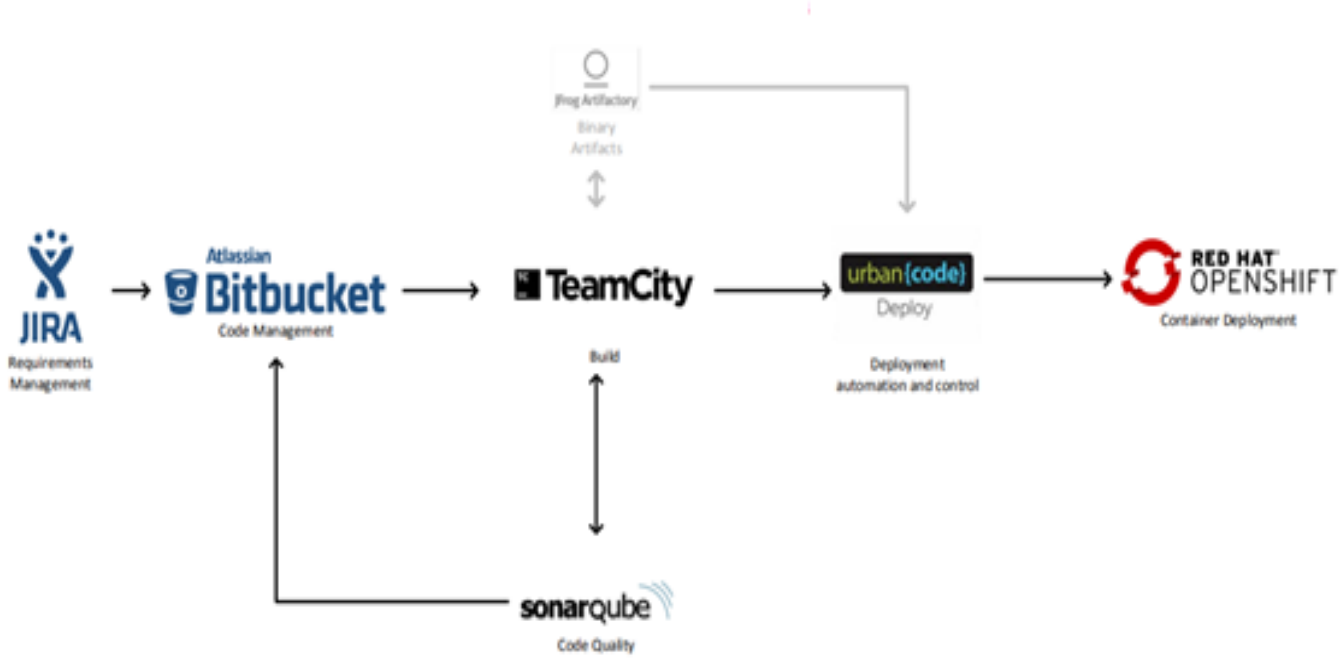
Brief about Docker, Kubernetes and OpenShift

- **What is OpenShift ?**
- OpenShift is a Platform-as-a-service(PaaS) application
- OpenShift offers the ability to deploy your web application code using a library of pre-defined platform images that build your environment
- So using OpenShift you can build and obtain your Docker images and deploy your application on top of them
- OpenShift is a layer on top of Docker and Kubernetes that makes it accessible and easy for the developer to create applications and a platform for operators to deploy containers on for both development and production workloads.

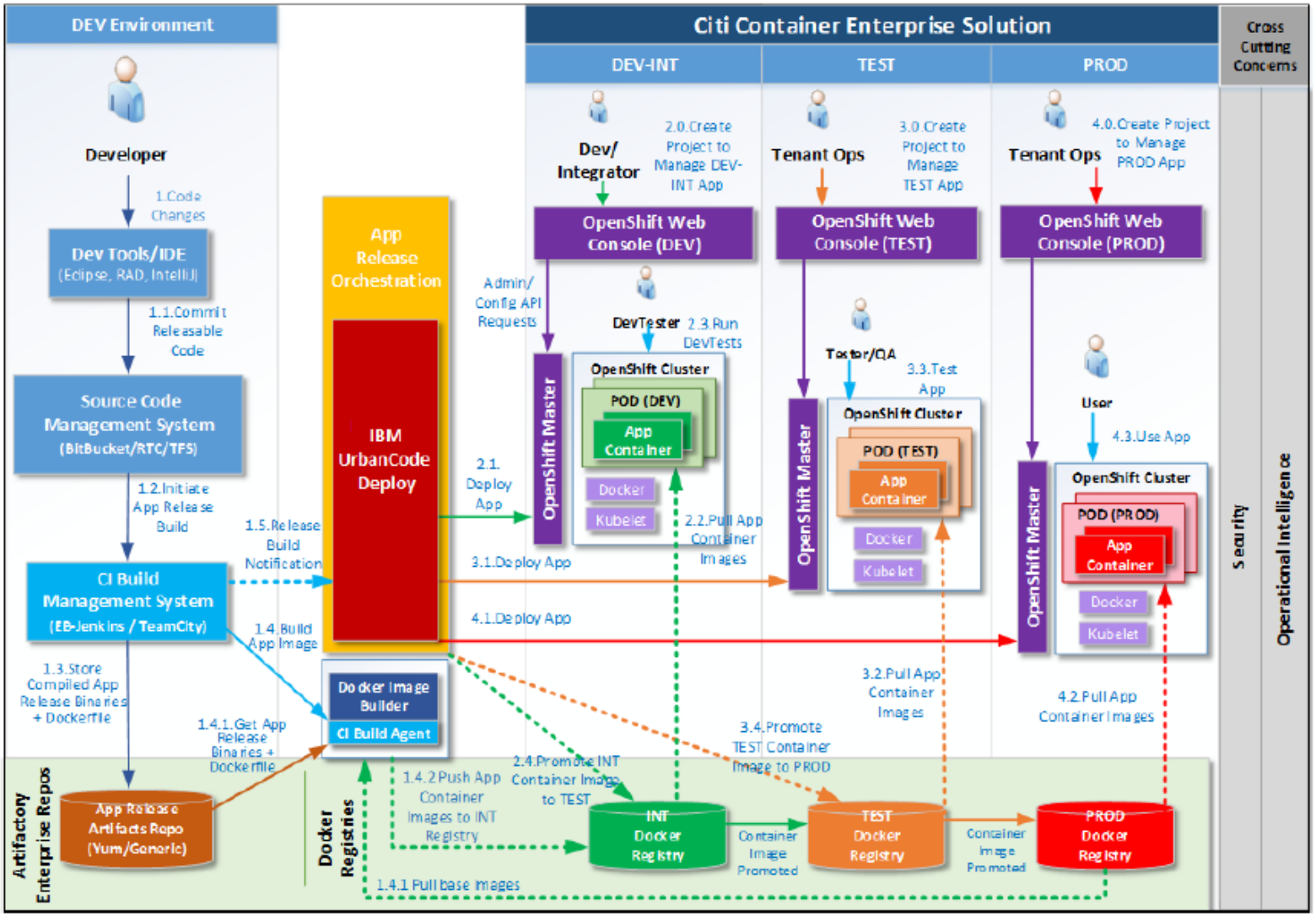
Features

- OpenShift has a visual web console that makes it easier for developers to perform the actions needed to deploy and run existing source code projects.
- Tasks like scaling the application containers, creating projects, viewing log files, viewing the memory, graphical representation of the application and CPU utilization of a container, and other common functions.
- The web console also has an integrated logging and metrics feature.

Single Click Deployment Flow on OpenShift



Enterprise Container Solution – CI/CD Architecture.



My POC Application: - (MMITrading – Ext JS & Java)

- **DEMO**

Detailed Steps to deploy application on OpenShift

- **Create the war file using Maven/Ant build and put it in the home directory of your BYOD and also put the below dockerfile in the same directory**.

- **Dockerfile :**

- FROM docker-enterprise-prod-local.artifactrepository.citigroup.net/cate-citicloud-tomcat/rhel7-tomcat:8.0.18_4
- *#copies the local file iswmmitrading.war , in this case it is in the same directory with Dockerfile, to the newly created folder in the image*
- COPY iswmmitrading.war /tmp
- RUN unzip /tmp/iswmmitrading.war -d $CATALINA_HOME/webapps/svs
- EXPOSE 20000

- **Build the docker image**
- docker build –t iswmmitrading:1.0.0

- **Capture the new generated image ID**
- docker images | grep iswmmitrading

Detailed Steps to deploy application on OpenShift

- **Use image if from above command output and run the image as a container using below command**
- docker run –d <imageId>

- **Use the generated container id to validate the iswmmitrading services, it will show below output if iswmmitrading services works fine.**
- docker exec <imageId> curl https://localhost:2000/index.html -k
- *If above command works it will show the output with Ok in the html tag*

- **Deploy application on OpenShift setup**
- oc new-app –docker-image=<image path> --name <app name you want to give>

- *Till now our app has been deployed to OpenShift and service (Kubernetes concept) has been created .*

- **Expose the service through a route to the external host.**
- oc create route passthrough –service=<service_name from previous command> --hostname=iswmmitrading.byo.$(hostname -f)

Detailed Steps to deploy application on OpenShift

- **In case you are sure about your Docker image, you can directly deploy your application to OpenShift using below commands**

- **Create the war file manually and put it in the home directory of your BYOD and also put the below dockerfile in the same directory**.

- **Dockerfile :**
- FROM docker-enterprise-prod-local.artifactrepository.citigroup.net/cate-citicloud-tomcat/rhel7-tomcat:8.0.18_4
- *#copies the local file iswmmitrading.war , in this case it is in the same directory with Dockerfile, to the newly created folder in the image*
- COPY iswmmitrading.war /tmp
- RUN unzip /tmp/iswmmitrading.war -d $CATALINA_HOME/webapps/svs
- EXPOSE 20000

- **Create a new application using Dockerfile .**
- oc new-app . –-strategy=docker - -name=<appname>

- **In this step we will start the build and if successful it will automatically deploy the application as well.**
- oc start-build <appname> --from-dir=./

- **Expose our application to the external host .**
- oc create route passthrough –service=<service_name> --hostname=iswmmitrading.byo.$(hostname -f)

Detailed Steps to deploy application on OpenShift

- **OpenShift deployments with Templates. And leverage secrets and configmap.**

- Templates are for creating or refreshing Deployment definitions. The logic for refreshing existing definitions is built in the oc client.
- You must have these three files which will be used

ishant-mmi-setup-secret.yaml.txt

service-mmi.json

parameter.json

- **Create the secrets :**
- oc create secrets special-config --from-literal=jdbc.mmi.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver --from-literal=jdbc.mmi.userid=profile --from-literal=jdbc.mmi.password=Welcome1 --from-literal=jdbc.mmi.url=jdbc:sqlserver://iswswdbsit03.nam.nsroot.net:2431;databaseName=profile

- **Verify whether the secrets has been created and verify they must be encrypted form:**
- oc get secret-name -o yaml

Detailed Steps to deploy application on OpenShift

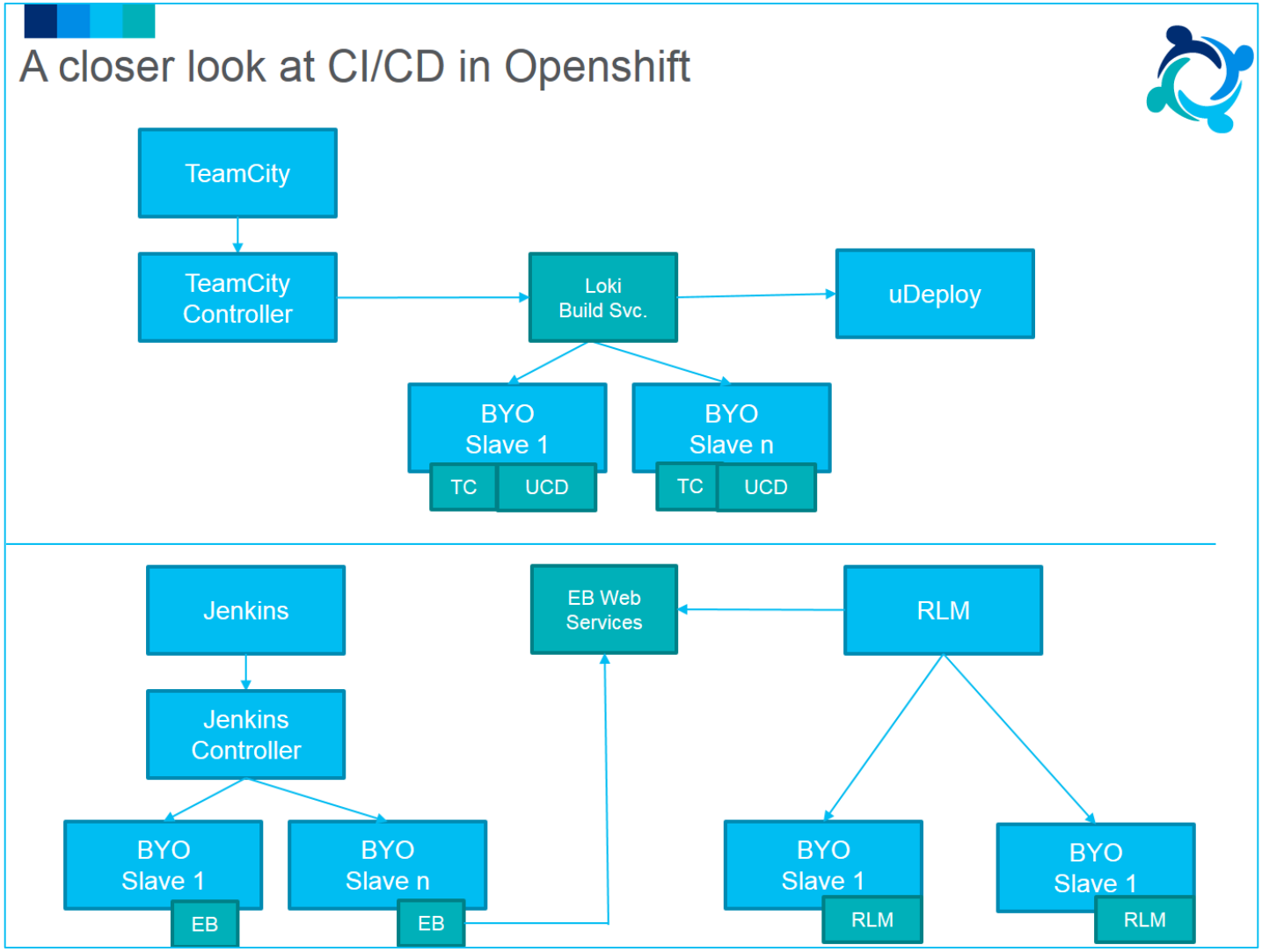
- **Create the new template , template contains all the information about your application.**
- oc create -f ishant-mmi-setup.yaml

- **Create the new deployement config , deployementconfig is configure where new parameter will be passed.**
- oc process -f ishant-mmi-setup-secret.yaml $(cat parameter.json)| oc apply -f –

- **Create the new service for our application .**
- oc create -f service-mmi.json

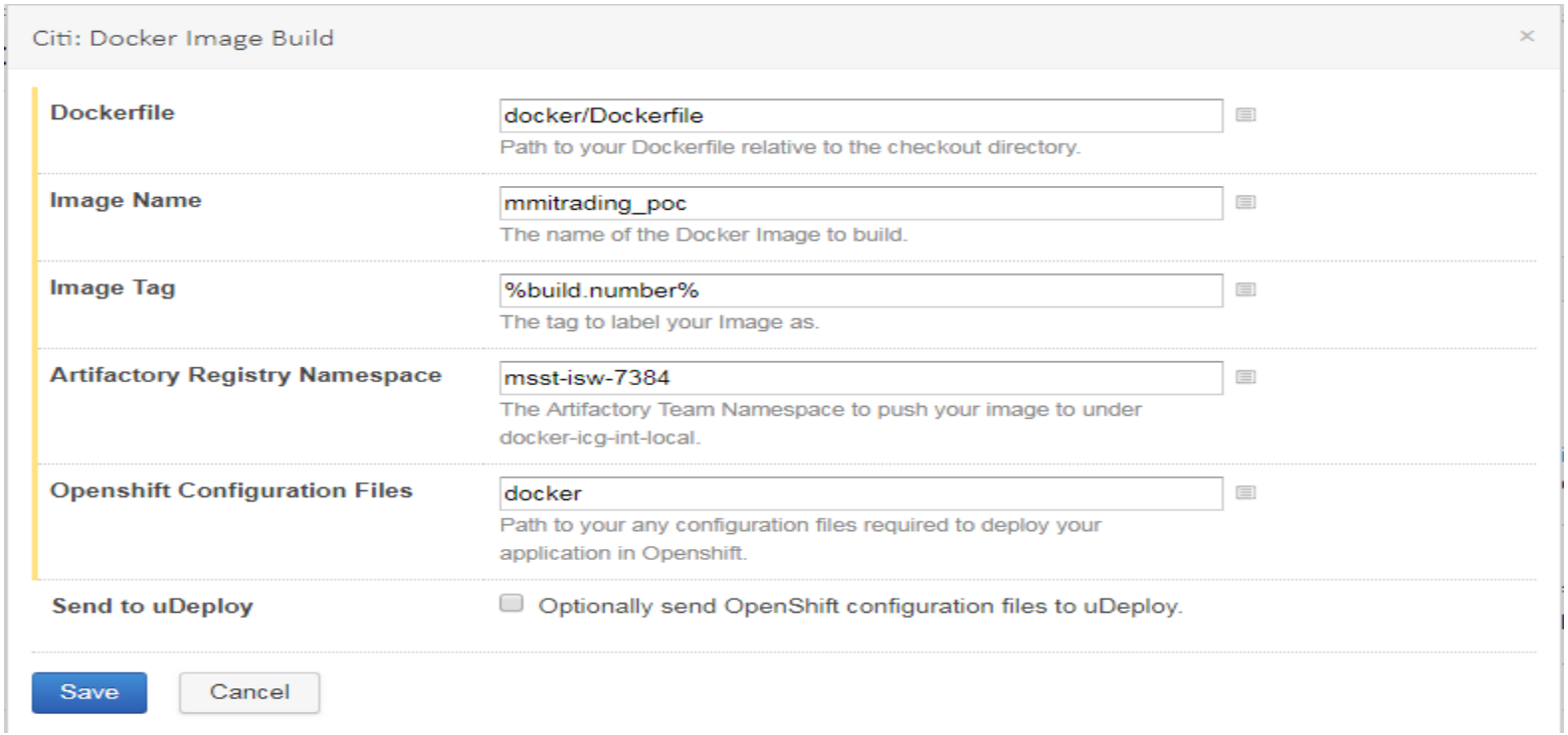
- **Create the route to expose our service to external host .**
- oc create route passthrough --service=secretexampleone --hostname=secretexampleone.byo.$(hostname -f)

Build Docker Image for OpenShift with ICG Build



Build Docker Image for OpenShift with ICG Build

- Build Image and push to Artifactory INT Docker registry. Optionally you can also enable upon completion of this step, a uDeploy delivery which will send the OpenShift configuration files to uDeploy. Finally, an additional optional step is available to trigger an auto-deployment upon successful completion of the above.



Build Docker Image for OpenShift with ICG Build

Citi: Send to uDeploy

Component Name

OpenShift_POC

Optionally override the component name to look for in uDeploy.

Component Id

163f7621-ee2e-1d6a-8e6a-86f65b62d7fe

Optionally set a specific uDeploy component ID.

uDeploy Environment

Production UCD1 - 6.2.1 (https://releasedeployment.ti.citigroup.net:8443)

Send to uDeploy UAT or Production

Branch Name

Branch Name:

Optionally specify the branches to send to uDeploy as a newline-delimited set of rules in the form of +|-:branch name (with the optional * placeholder)

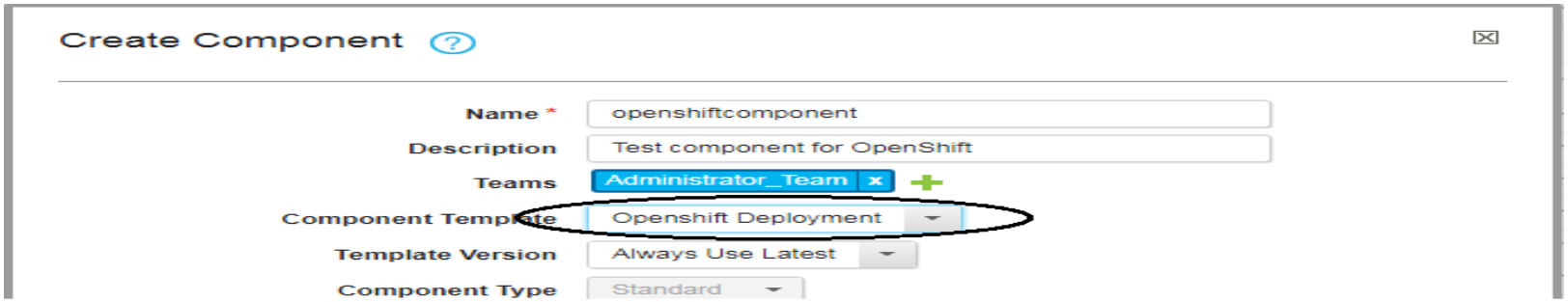
Enable Auto Deployment

☐ Trigger a uDeploy deployment after upload.

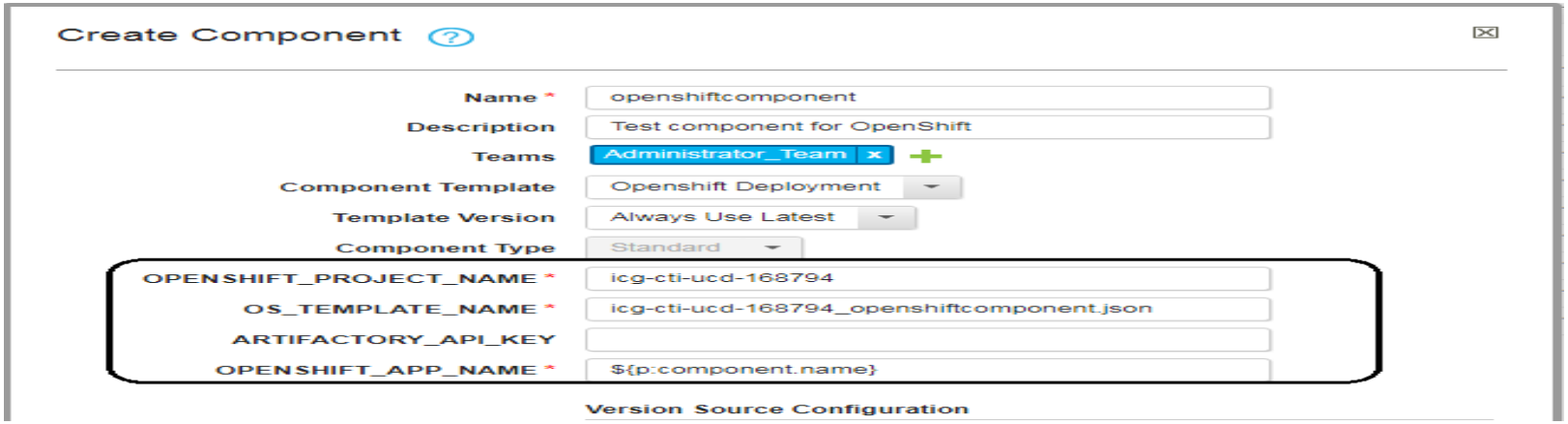
Save Cancel

Udeploy Configuration to deploy images

- Create new Component in the Udeploy and select "OpenShift Deployment " component template.



- Enter the Component level properties values in the respective fields and save.



Udeploy Configuration to deploy images

- Define the values for the Component environment properties in the respective application environments.

Resources | History | Calendar | Configuration | Changes

Basic Settings
Environment Properties

Environment Properties

Version 1 of 1

Add Property | Batch Edit

Name	Value
ENV.CATEGORY	DEV
ENV.REGION	

2 records - Refresh Print

Component Environment Properties

Filter By Component: openshiftcomponent

Version 2 of 2

Use Batch Edit Mode

OPENSHIFT_URL *	https://ose3gtdc1-master.nam.nsroot.net:8443
ART_SOURCE_REPO *	docker-icg-dev-local
ART_TARGET_REPO *	docker-icg-int-local
OPENSHIFT_FID *	FID
OPENSHIFT_FID_PASSWORD *	••••
OS_DEPLOY_CONFIG	${p:OPENSHIFT_APP_NAME}
OS_PARAMETER_FILE *	icg-cti-ucd-168794_openshiftcomponent.int
OS_PATCH_FILE	
DOCKER_REPOSITORY *	docker-icg-int-local/msst-rates-ucdtest1-168794

Question & Answer ?

Thankyou ☺