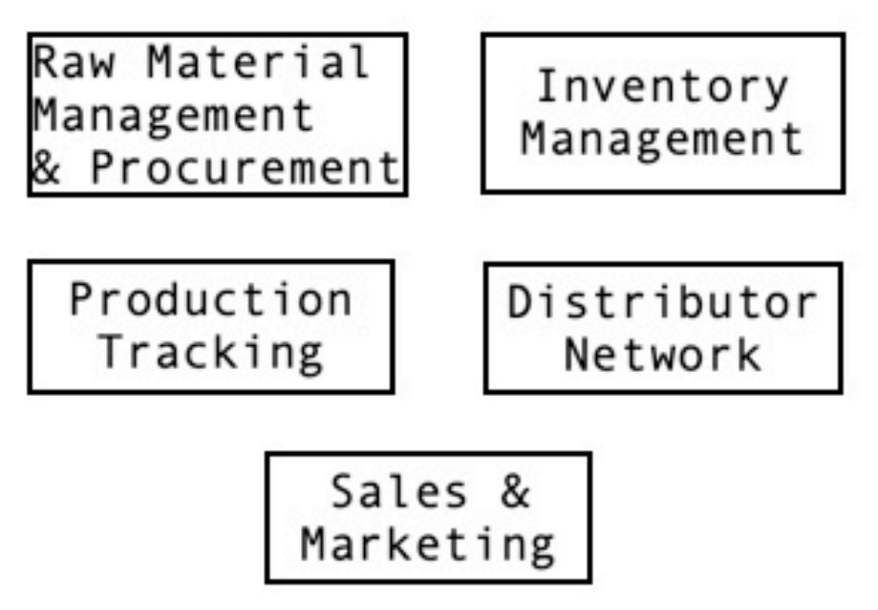
Can we leverage jee platform capabilities/features for developing and delivering microservices architecture based applications? If not how does the microservices applications are developed and delivered in the realworld?

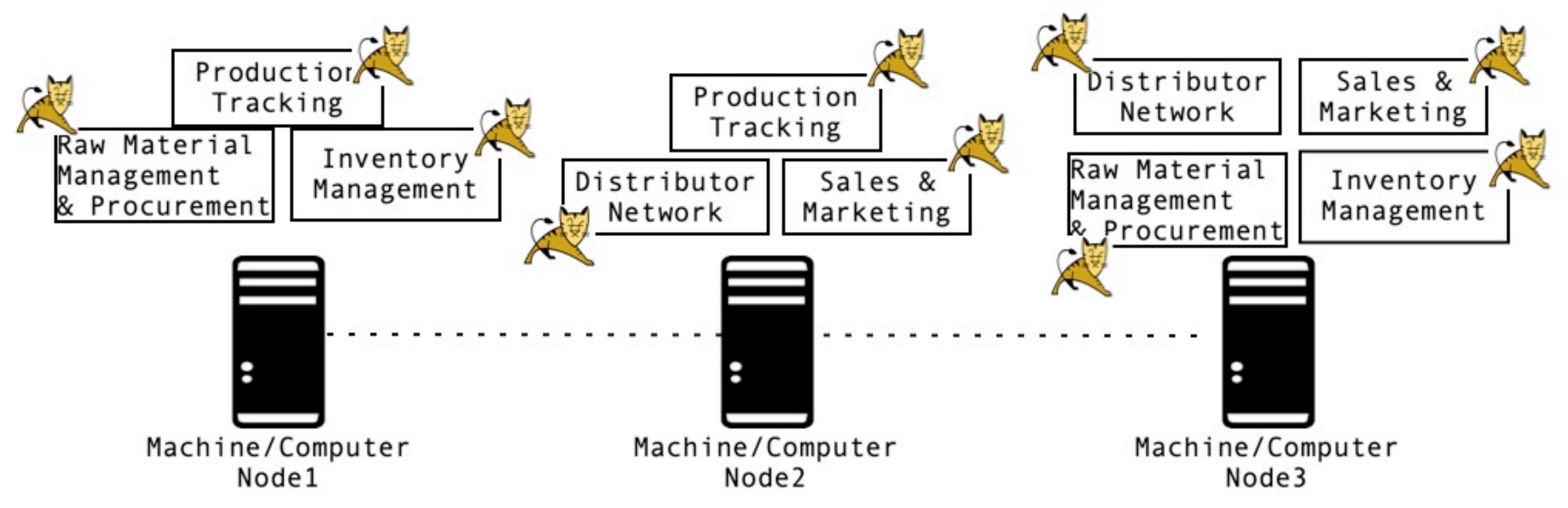


5 Microservices are built into 5 independent projects, build , packaged separately and deployed independently

How to develop and deploy the Microservices applications?
1. Each Microservice application is built out of its own sourcecode project without any classpath references across them, build, packaged and deployed independent of each other

2. all the microservices are not deployed on one single machine, each microservice should be deployed on its own server (jvm) due to below reasons:
  1. load on one microservice should not affect the others
  2. fault isolation, if one microservice has been crashed it should not impact the others
  3. release management: upgradation, patching a microservice should not effect the availability of others.

3. Atleast #2 instances of a microservice application should be running on #2 different nodes of the cluster for high-availability
4. We can run more than 1 microservice on a machine, if the node has sufficient computing resources.



Incase of Monolithic Application, the same application will be deployed across all the nodes of the cluster, so
1. we need domain configurations (managed servers, libraries, security roles/users)
2. resources (datasource, jms queues/topics)
3. application being deployed across all the nodes
so jee application server helps us in deploying, propating the resources and configurations across all the notes of cluster

whereas when it comes to microservices architecture based applications, different microservices are deployed on different nodes of the cluster, not every node has all the microservices deployed and running. So that
1. different nodes required different domain configurations
2. different resources
3. not all the microservices are running across all the nodes

But when it comes to JEE application server, it takes care of deploying the application, configurations/resources across all the nodes of the cluster which is not applicable for a Microservice application.

In-Short:
1. JEE Cluster is symmetric, all the nodes of the cluster looks alike (same)
2. Microservices doesnt hold the concept of Application Cluster, because each node runs with different microservice applications that requires different configurations/resources. so the concept of cluster computers is Asymmetric

Hence we cannot leverage JEE Server deployment/monitoring capabilities in delivering and running Microservice applications on a Cluster of Computers.

From the above we can understand we should not use JEE Application Server capabilities for managing in running microservices applications on a Cluster Infrastructure. But we still need Enterprise Class-Level features in developing the microservices applications
1. security
2. resource pooling (datasource connection pooling)
3. caching
4. global transactions
5. jms messaging
6. auditing
7. schedulers
9. application logging
10. monitoring
11. load balancing
etc
Looks like all of these enterprise class-level features are required for microservices applications as well and these are offered by the jee servers/jee apis, so can we use still jee platform services for building the microservices with these enterprise class-level features?

No, dont use JEE platform for building enterprise class-level features for microservices, Why?
Incase of microservices, not all the microservices requires all the enterprise class-level features. Different Microservices requires different set of features based on their nature.
For eg.. Microservice A : requires schedulers and caching capabilities
         Microservice B : requires messaging services & resource pooling

In this way different microservices requires different cross-cutting capabilities or features to be used based on their nature of functionality. But when it comes to JEE Application Servers, these all cross-cutting enterprise features are pre-packaged and pre-built aspart of the server infrastructure itself. So upon starting an jee container, all of these cross-platform capabilities will be booted up and requires huge computing resources in keeping these servers running due to which jee containers are considered as heavy weight servers.

So if we deploy microservices on JEE infrastructure, we endup in having huge infrastructure for running the jee containers, even though we dont use most of the cross-platform services offered by the jee.

The JEE containers are not designed to be light-weight in nature, we cannot configure or create manager servers in jee container to have few platform services being entitled. so instead of using jee containers/platform services to build enterprise class-level features into microservices we can leverage third-party libraries in building these enterprise capabilities and these are pluggable/configurable and seems to be more flexible than jee platform

1. Connection Pooling = dbcp, c3p0, proxool, hikari etc
2. Messaging = MQ Series, Active MQ, Kafka, Rabit MQ etc
3. Security = Spring Security
4. Transactionality = every persistence framework vendor supports transactionality out of box, and we can use Spring Transaction module for implementing declarative transaction management.
5. Caching = Distributed Caching libraries like EHCache, SwarnCahe, JCache, OSCahe. Cache Infrastructure: Redis, MemCache etc
6. Logging = Plenty of logging libraries are available.

Based on the requirements of the microservices, we can choose and pick any of these third-party libraries in integarting and build the enterprise class-level features to our application. It would be more easy to build microservices applications through the help of Spring Framework, Spring Boot and Spring Cloud as they have integrations with most of these third-party libraries.

Load Balancing:
Load balancer distributes the incoming http traffic across the nodes of the cluster assuming all the nodes are symmetric or our application is running across all the nodes of the cluster.

But when it comes to Microservices application deployment, different nodes are running with different microservices, so loadbalancer dont have the knowledge of which microservices are running on which nodes, so it cannot distribute the incomming traffic for a microservice application to an appropriate node, so we dont need loadbalacer capabilities provided by JEE Server.