**Control Plane:** The components of the Anypoint Platform architecture that you use to design, deploy, and manage APIs and Mule applications. The control plane contains product features and components that are part of the Anypoint Design Centre, Anypoint Management Centre and Anypoint Exchange.

What is getting stored in the Control Plane?

Stores Information (metadata)

API analytics information like how many times API is called, how much time taken to complete the transaction

JVM Utilization

CPU Utilization

Jar files

Logs (depends on types of deployment)

Control Plane Options

1. US Cloud

Mulesoft hosted control plane.

Physically hosted within United States

2.EU Cloud

Mulesoft hosted control plane

Physically hosted within European Union

3.Mulesoft Government Cloud

Managed by Mulesoft and is suitable for government agencies

4.Customer-Hosted Control Plane

Customers can host control plane within their own data centre.

**Runtime Plane:**

The components of the Anypoint Platform architecture to which your APIs and Mule applications are deployed and made available to your users. The runtime plane includes the Mule runtime server and supporting services.

Applications are deployed to runtime plane

All data, payloads are processed in runtime plane.

**Runtime Plane options**

1.CloudHub

CloudHub enables you to deploy an API or a Mule application on a platform that’s managed by MuleSoft.

2.Runtime fabric

Runtime Fabric is a container service that enables you to run Mule applications and API gateways within a data centre or third-party cloud environment that you control and manage.

3.Standalone Runtimes

The standalone option enables you to host Mule runtime engine server and related services in an environment that you manage

**Network latency: Network latency**, sometimes called lag, is the term used to describe delays in communication over a **network**. **Latency** meaning in **networking** is best thought of as the amount of time it takes for a packet of data to be captured, transmitted, processed through multiple devices, then received at its destination and decoded.

**Throughput: Throughput** is the rate of production or the rate at which something is processed. When used in the context of communication networks, such as Ethernet or packet radio, **throughput** or network **throughput** is the rate of successful message delivery over a communication channel.

**CloudHub:** Cloudhub is an Integration Platform as a Service (iPaaS). It enables us to deploy and run the application in the cloud via Runtime Manager. CloudHub is a scalable, multi-tenant, elastic, secure, and highly available iPaas. CloudHub is managed via the Runtime Manager console in the Anypoint platform. It enables us to deploy Mule applications from Anypoint Studio using the CloudHub API or the CloudHub Command Line Interface.

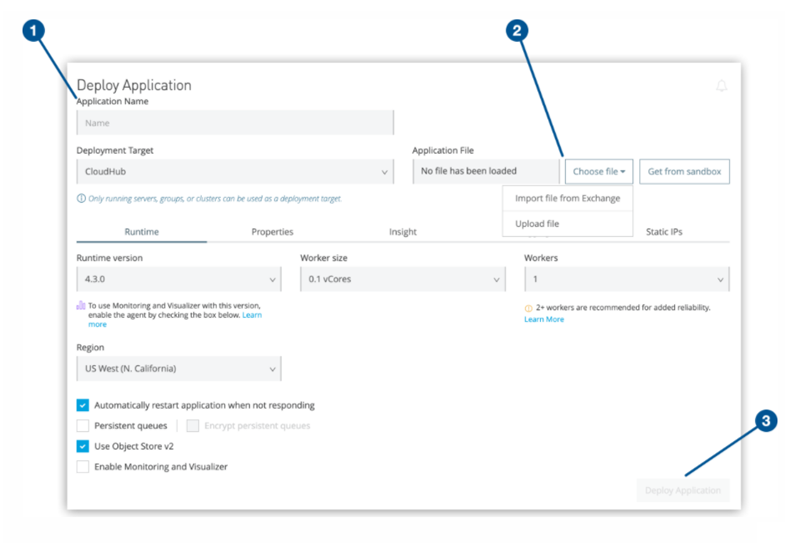
**2.1.2 Deployment of Mule applications to CloudHub:**

**Anypoint Runtime Manager:**

To deploy from Runtime Manager to CloudHub:

1. Sign in to Anypoint Platform.
2. Select Runtime Manager.
3. In the Applications page, click Deploy application.

We can see the Deploy Application page:



**Anypoint Studio:**

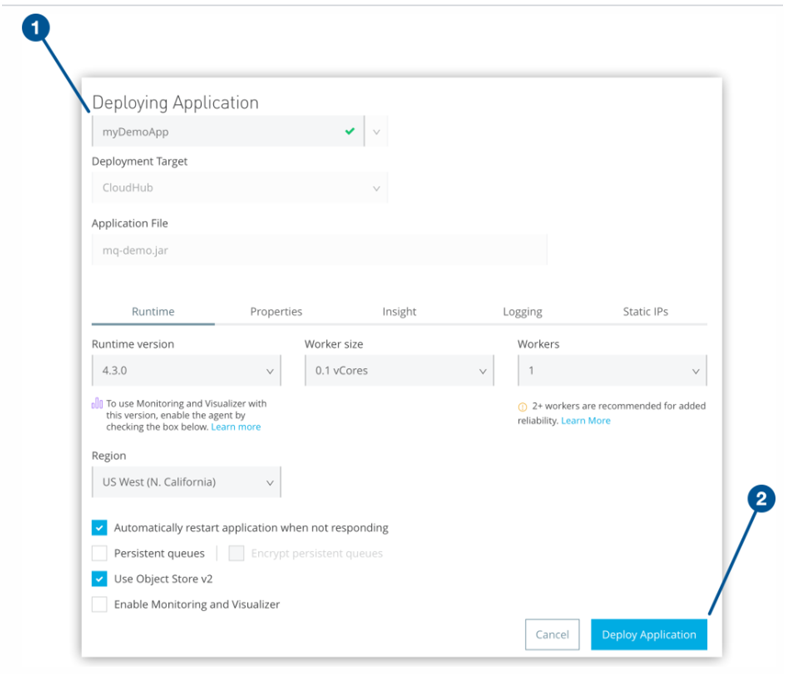
Deploy an Application from Studio

We can deploy our applications directly from Studio to CloudHub. This option is helpful if we want to test our application while developing it.

*To deploy from Studio to CloudHub:*

1. Open application in Studio.
2. In Package Explorer, right-click the project folder and select Anypoint Platform > Deploy to CloudHub.
3. If this is our first time deploying from Studio, provide your Anypoint Platform credentials at the prompt.
4. Studio stores our credentials for future use.
5. We can manage these credentials by selecting Anypoint Studio > Preferences > Anypoint Studio > Authentication.

After we sign in, we see the Deploying Application page:



**Advantages of deploying applications to CloudHub:**

* Hastle free deployment
* Zero downtime upgrades
* Scalability
* Access to OSV2, Anypoint MQ and Persistent Queue

**Disadvantages of CloudHub deployment model:**

* Domain projects are not supported
* Workers could be expensive
* VPC and VPN charges are extra

**What is a Shared Load Balancer (SLB):** CloudHub provides a default shared load balancer that is available in all environments. The shared load balancer provides basic functionality, such as TCP load balancing. Shared load balancers don’t allow you to configure custom SSL certificates or proxy rules.

Additionally, shared load balancers have lower rate limits that help ensure platform stability. MuleSoft regularly monitors and scales these limits as necessary. Rate limits on shared load balancers are applied according to region. If you are deploying an application to workers in multiple regions, the rate limit for each region might be different.

If an application exceeds the rate limit for a shared load balancer, the load balancer returns a 503 Service Unavailable response. To perform custom load balancer configuration or have higher rate limit thresholds, you must use a dedicated load balancer. The shared load balancer supports TLS versions 1.1 and TLS 1.2.

**EC2 instance:**

An **EC2 instance** is a virtual server in Amazon's Elastic Compute Cloud (**EC2**) for running applications on the Amazon Web Services (**AWS**) infrastructure. **AWS** is a comprehensive, evolving cloud computing platform; **EC2** is a service that allows business subscribers to run application programs in the computing environment.

**Hybrid Model:**

In this we are having mulesoft managed control plane and customer managed runtime plane.

To use the hybrid option, we first register our Mule servers with the Runtime Manager agent. Then, from Runtime Manager, we can optionally add those servers to server groups or clusters to provide high availability. Finally, we deploy our applications from Runtime Manager to either a server, server group, or cluster.

The following features and limitations apply when using Runtime Manager to manage hybrid deployments:

**High availability**

To achieve high availability in hybrid deployments, use server groups or clusters. Clustered Mule instances have distributed shared memory, which provides persistent VM queues, transactions, and cluster-wide data storage.

**JDK versions**

For hybrid deployments, it is necessary to see the release notes of the Mule (standalone) runtime engine to determine the minimum JDK supported version.

**Load balancing**

Load balancing is not provided for hybrid deployments. We can manage load balancing with the tools connected to our on-premises resources.

**Logging**

The Runtime Manager logging feature is not available for hybrid deployments. We can configure on-premises apps to send data to external analytics tools, such as Splunk or ELK, to manage our logs.

**Monitoring**

The Runtime Manager cloud console includes dashboards that provide performance metrics for all deployed applications.

With Runtime Manager, we can set up both alerts based on alert conditions on deployed apps and standard notifications to inform us when events occur in our applications. We can also create alerts that are triggered by events related to the servers on which they run, such as reaching a certain CPU usage threshold or adding a new node to a cluster.

**Object store**

The object store infrastructure is not available for hybrid deployments. To use object stores, we must configure a database to store data.

**Scheduling**

The scheduling feature is not available for hybrid deployments. To schedule tasks, use the Scheduler endpoint element in your flows.

**Security updates**

After an application is deployed and running, we must apply any security updates for the selected runtime version manually.

**Troubleshooting**

CloudHub provides the Insight troubleshooting tool that provides in-depth visibility into business transactions and events on your Mule apps deployed through Runtime Manager.

### **Clustering with MuleSoft Anypoint Runtime Manager**

A cluster is a set of up to eight servers that act as a single deployment target and high-availability processing unit. Unlike in server groups, application instances in a cluster are aware of each other, share common information, and synchronize statuses. If one server fails, another server takes over processing applications. A cluster can run multiple applications. In simple terms, virtual servers composed of multiple nodes and they communicate and share information through a distributed shared memory grid.

By default, Mulesoft ensures the High availability of applications if clustering is implemented. Consider a scenario where one of the nodes in the cluster crashed or goes down and is under maintenance. In such cases, Mulesoft will ensure that requests are processed by other nodes in the cluster.

Clustering is only supported by on-premise Mule runtime and it is not supported in Cloudhub.

Before creating a cluster, we must create the Mule runtime engine instances and add the Mule servers to Anypoint Runtime Manager.

**Types of Clustering**

While configuring a cluster, we need to specify either unicast or multicast options for identifying a node within the cluster:

* **Unicast**

A unicast cluster requires that we configure the IP addresses of the nodes in the cluster. If a server has multiple interfaces, use the internal IP address that allows the node to communicate directly with other nodes. Clustering across different subnets is not supported.

Because Mule relies on the IP address to identify the server, IP addresses can’t be dynamically assigned using DHCP for servers on a unicast cluster. If a server is restarted and uses DHCP to get a new IP address, we must add the server to the cluster using its new IP address.

The server status must be Running to be added to a unicast cluster.

* **Multicast**

A multicast cluster comprises servers that automatically detect each other. Servers that are part of a multicast cluster must be on the same network segment.

One advantage of multicast clusters is that the server status doesn’t need to be ‘Running’ to configure it as a node in a cluster. Another is that we can add nodes to the cluster dynamically without restarting the cluster.

It is necessary to check with the network administrator to determine if multicast is allowed within our network, because many networks block multicast functionalities.

**Benefits of Clustering**

* Clustering helps in improving application performance by dividing the workload between nodes in the cluster.
* Horizontal Scaling can be done easily whenever required.
* Automatic coordination of access to resources, such as files, databases, and FTP sources.
* Automatic load balancing of processing within a cluster.
* Improve Performance, Automatic Failover, High Availability.

### **Shared Resource Support using Mule Domain Project**

Using the Mule Domain Project, the developer can connect multiple Mule applications together, allowing them to share resources and connector versions, which is an approach impossible to deploy while using Cloudhub.

When we deploy Mule on premises, we can define global configurations such as default error handlers, shared properties, scheduler pools, connectors, and connector configurations to be shared among all applications deployed under the same domain. These resources are known as shared resources. To host them, we must create a Mule Domain Project and then reference it from each of the projects that use the elements in it. Once defined, any Mule app associated with a particular domain can access resources in this file. Note that Mule apps can be associated with only one domain at a time.

Shared resources provided by Mule Domain Project also allow Mule Services to use VM Queue as a method of communication between one another, using internal server memory instead of HTTP traffic – which is more secure, faster and allows preserving processed data if servers are in a cluster.

Shared resources allow multiple development teams to work in parallel using the same set of reusable connectors. Defining these connectors as shared resources at the domain level allows the team to:

* Expose multiple services within the domain through the same port.
* Share the connection to persistent storage.
* Share services between apps through a well-defined interface.
* Ensure consistency between apps upon any changes because the configuration is only set in one place.

To share the metadata, keep the Mule domain project open in Anypoint Studio. Otherwise, you must enter the metadata manually on the linked Mule projects.

**Use Case**

Generally, when we deploy the application having HTTP Listener on mule standalone run time, each application needs to be deployed on different ports otherwise we will get an error.

To avoid this, we can implement the domain project and define HTTP Listener connection with some port (i.e. 8081) and that domain project can be referred across all the other projects or applications so we can run all the applications on the same port (i.e. 8081).

### **Disadvantages of Hybrid Implementation**

A hybrid-implementation approach, even using domain-project based applications, also has several disadvantages.

**More complicated design of Deployment Pipeline and Release Procedures**

Mule Domain Projects cannot be deployed through Anypoint Platform Runtime Manager – packages containing domain projects need to be deployed via SSH directly to a Server.

If we want to achieve auto-scaling, we need to design Deployment Pipeline to take care of connecting servers we provide to Anypoint Platform Cluster.

**Reduced out-of-the-box monitoring**

Another disadvantage not related to Deployment pipeline is reduced out-of-the-box monitoring for Mule services and servers. The Cloudhub server, being fully-provisioned by Mulesoft, cannot be accessed via SSH, so all necessary metrics, logs and alerts are possible to be easily implemented on Anypoint Platform. Hybrid implementations can be provisioned on different server configurations, with different operational systems, and have the Anypoint Platform monitoring reduced to minimum (like the number of errors for services, response time, or CPU usage or heap memory for servers). For more advanced monitoring, there is an additional development on Anypoint Platform to be done, or even a decision to be made concerning choosing a different monitoring platform.

### **Load Balancing**

Load balancing refers to efficiently distributing incoming network traffic across a group of backend servers, also known as a server farm or server pool.

A load balancer acts as the “traffic cop” sitting in front of our servers and routing client requests across all servers capable of fulfilling those requests in a manner that maximizes speed and capacity utilization and ensures that no one server is overworked, which could degrade performance.  If a single server goes down, the load balancer redirects traffic to the remaining online servers. When a new server is added to the server group, the load balancer automatically starts to send requests to it.

**Types of Load Balancing**

Load balancers typically come in two flavors: hardware‑based and software‑based. Vendors of hardware‑based solutions load proprietary software onto the machine they provide, which often uses specialized processors. To cope with increasing traffic at our website, we have to buy more or bigger machines from the vendor. Software solutions generally run on commodity hardware, making them less expensive and more flexible. We can install the software on the hardware of our choice or in cloud environments like AWS EC2.

Mule on-premises standalone servers already support clustering where workload is divided among the different server nodes in the cluster. However, it requires a software load balancer for distributing the incoming HTTP request across its nodes which is exposing HTTP applications.

**Benefits of Load Balancing**

* Reduced Downtime
* Scalable
* Redundancy
* Flexibility
* Efficiency
* Global Server Load Balancing

### **Security for Hybrid Deployment**

By default, only metadata pushed by the Runtime Manager agent in each Mule runtime engine flows to the cloud. No application data is exposed.

The agent monitors and controls Mule, and publishes its data to the control plane. We can control Mule from external systems by calling Runtime Manager agent APIs, or we can specify that Mule publishes its data to external systems.

We can change the default behaviour so that the agent pushes IDs, final average numbers, or any data we find useful for monitoring or keeping control of applications.

**What is Runtime Fabric?**

Runtime Fabric (RTF) is a container service to deploy mule runtime on cloud (AWS / Azure) or on a data center (on-premise). It provides all cloud (PaaS) benefits such as high availability, automatic failover, rolling upgrade, etc. RTF is based on Docker containers and Kubernetes for orchestration and has been packaged/maintained in such a way that very little or no knowledge of docker is required to run RTF.

With the help of RTF we can deploy multiple versions of mule-runtime. RTF helps to achieve consistency and portability with the support of container service. We can deploy our API/Integration solution across many platforms like AWS, AZure or on-premise data centers. Runtime fabric automates the deployment and orchestration of Mule application and API gateway.

RFT is based on docker and kubernetes to achieve portability and horizontal scalability. It includes all the components required to run the mule application. We can manage the from our Anytime platform ( control plane).It provides the same feature as Cudhub with added benefits of our own deployment on AWS / AZure.

## We can horizontally scale the application with multiple instances running . There is an internal load balancer available that helps to balance the load across all running instances. While deploying the application to runtime fabric it provides the  option to select runtime version , number of replicas , vCPU and memory.

### Why RTF?

There are many reasons customers opt for RTF as their deployment strategy:

* Multicloud support
* Application Isolation
* Support for Multiple Runtimes
* Zero downtime upgrades, rollbacks, and scaling
* High availability
* Minimal infrastructure skills required

#### Multicloud

RTF can be installed to any cloud or customer datacenter, and managed via the AnyPoint Platform Runtime Manager

**Application Isolation**

The underlying Docker container technology provides strong isolation through the use of Linux [kernel namespaces and control groups](https://docs.docker.com/engine/security/security/). This increases the reliability and security of your application network by isolating any potential failures or compromises in a way that is very difficult to reproduce in traditional customer-hosted deployment scenarios. All of this is transparent to the administrator and there is still full support for the AnyPoint Platform concepts of [Environments](https://docs.mulesoft.com/access-management/environments) and [Business Groups](https://docs.mulesoft.com/access-management/business-groups).

#### Multiple Runtimes

Because of the above-mentioned isolation, it is also possible to deploy multiple versions of the Mule runtime to the same RTF cluster. Customers can run support Mule 3 and Mule 4 applications on one infrastructure and channel their efforts toward creating value for the business rather than maintaining multiple environments.

#### Zero Downtime Upgrades, etc.

Orchestrating zero downtime upgrades, rollbacks, and scaling of Mule applications is not trivial. While this has always been available for CloudHub customers, now customers can enjoy the same painless automation in any cloud.

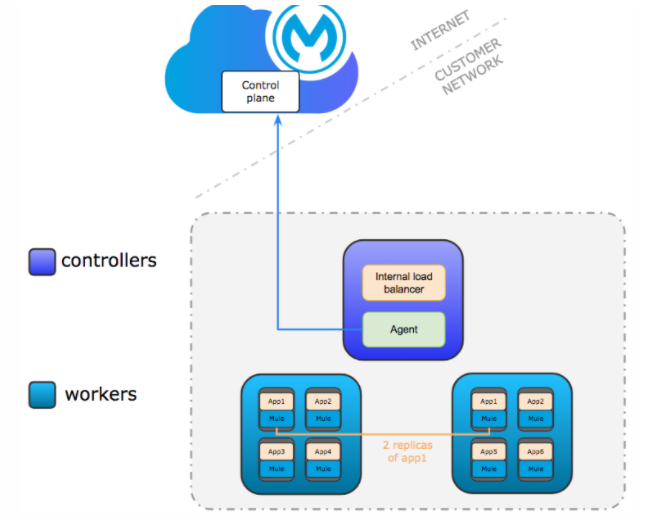
#### High Availability

RTF cluster installations are highly available by design, and deploying Mule applications in a highly available replica set or cluster is simple as moving a slider and clicking a checkbox.

#### Minimal Infrastructure Skills

Thanks to the packaging of RTF as a [Gravity appliance](https://gravitational.com/gravity/), installation, upgrades, and other cluster maintenance is trivial and the underlying Docker and Kubernetes technologies are mostly transparent to the end user. However, if required administrators can still dive deep with access to [Ops Center](https://docs.mulesoft.com/runtime-fabric/1.4/using-opscenter), [gravity CLI](https://gravitational.com/gravity/docs/cluster/#gravity-cli-tool), and [kubectl](https://kubernetes.io/docs/reference/kubectl/overview/).

**Runtime Fabric Architecture**

****

The architecture of the Runtime Fabric cluster is composed of two types of nodes: controllers and workers.

Runtime fabric is a collection of VM that work together to form a cluster. Each VM in the cluster serves as either a **“Controller”** node or **“Worker”** node

**Controller Node**

The RTF controller is in fact a Kubernetes controller, so the orchestration, distributed database, and internal load-balancing capabilities are still being leveraged here. In addition, most control plane communication with AnyPoint Platform also happens here.

The biggest consideration when deciding on the number and size of controllers is redundancy and load-balancing capacity. Requests to apps running on the worker nodes whether sourced from inside or outside the cluster will always go through the load-balancer on the controller nodes, ensuring the same clustering and zero-downtime upgrades, rollbacks, and scaling that CloudHub customers enjoy. However, keep in mind that if there are multiple controllers, an additional external load-balancer will be required for traffic from outside the cluster.

**The current maximum number of supported controller nodes is 5**

**Worker Node**

The RTF worker node is where regular Mule applications will get deployed as pods. The more workers you have, the more highly-available and performant your cluster will be.

**The current maximum number of supported worker nodes is 16.**

Runtime Fabric offer two types of deployment

* Development
* Production

These deployment options differ in the number of Controllers / Worker required .  Only the controller runs the Internal load balancer and agent to connect with “Control Plane”.

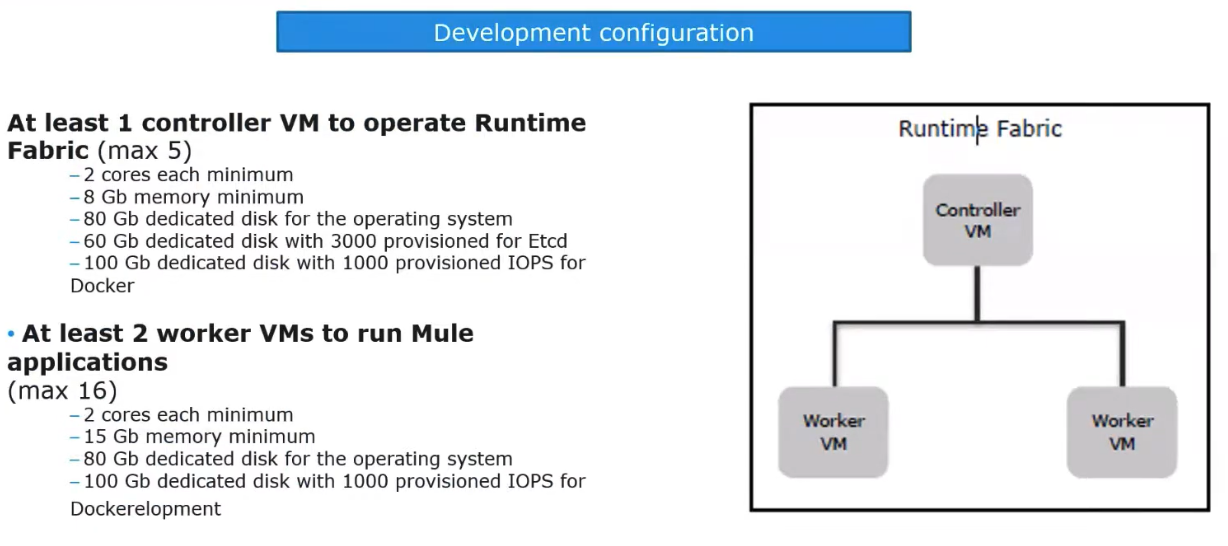
Recommended number of controller nodes for development and production are as

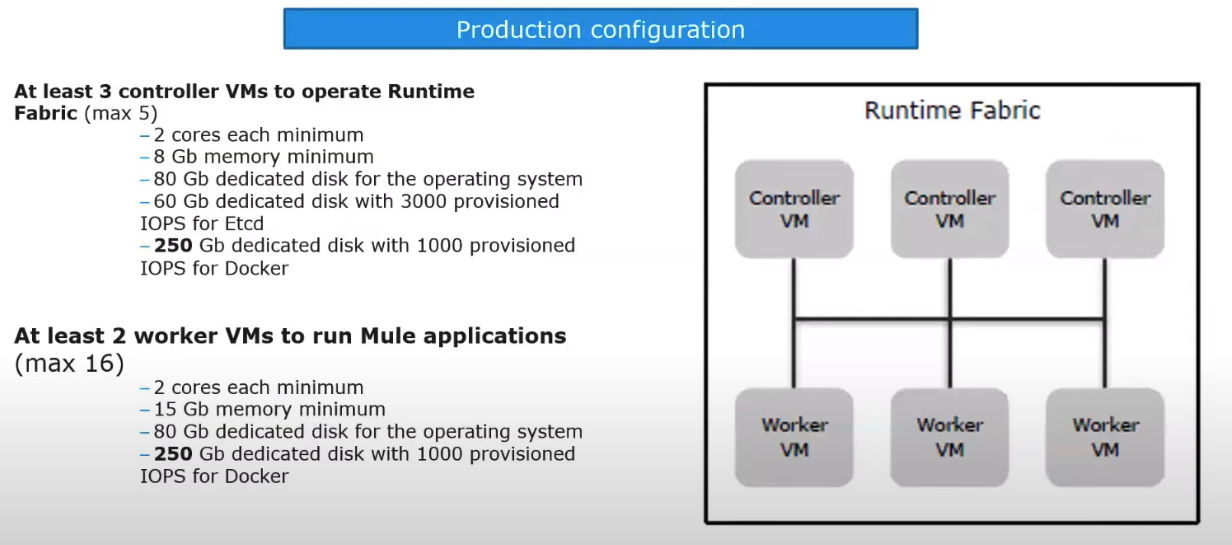
* Production – 3
* Development – 1

Worker nodes

* Production – 3
* Development – 2

These are done to achieve HA. This is necessary to achieve fault tolerance and high availability.





### **RTF Prerequisites**

We can use RTF if you have AnyPoint Platform Platinum or Titanium plans, and you can install it to any infrastructure that can run RedHat or CentOS Linux and meets the resource requirements. That includes local VM or bare metal installations, as well as all the leading PaaS providers (Amazon Web Services, Microsoft Azure, Google Cloud, Alibaba Cloud, etc.). However, MuleSoft currently only produces automated installation scripts for AWS and Azure and GCP in the near future.

<https://docs.mulesoft.com/runtime-fabric/latest/install-prereqs>

RTF can be setup by:

* using terraform script on AWS platform  ()
* using terraform script on Azure platform
* By doing it manually

### **Why PCE?**

If an organization has a strict compliance law on data processing within their own data centers then PCE is suitable. In such a case the organizations won’t even allow the flow of meta data outside the premise. Example industries such as banks, insurance corps, healthe care and government organization have scenarios where they want to more tightly manage their data.

## 2.5.1. Available Components

|  |  |
| --- | --- |
| Available Components |  |
| Anypoint Runtime Manager | Enables you to track and manage your deployed Mule applications. |
| Anypoint API Manager | Enables you to manage your registered APIs. |
| Anypoint Exchange | Enables your organization to share various assets for reuse across integration projects. |
| Access Management | * - Enables to configure User Management using LDAP * - Enables to configure SMTP server to send API Manager from PCE * - Set certificates and private key to handle SSL encryption * - Add customized disclaimer to the anypoint platform log-in message |
| Anypoint Design Center, excluding Flow Designer |  |

### **2.5.2. Supported Configurations**

To ensure platform performance, stability, and high availability, Anypoint Platform Private Cloud Edition (Anypoint Platform PCE) supports two network configurations: 3-node and 6-node.

1. **3-Node Configuration**

Three is the minimum number of nodes that can enable high-availability and failover. In this configuration, each node hosts the platform applications and services. You must configure a load balancer to use round-robin distribution of traffic among each of the three nodes

Each node also hosts an instance of the database and object store. Hosting the database and object store enables persistence but requires larger minimum disk and memory requirements. Although each node contains a database, only one database is used as the master. Applications on each node write to this database only. The other two database instances are hot standby instances of the master database that take over as the master database in case of failure.

1. **6-Node Configuration**

In the 6-node configuration, three nodes are dedicated to hosting platform applications and services. The other three nodes host the database and object store instances. Although each node contains a database, only one of the database nodes is used as the master. Applications on each node write to this database only. The other two database instances are hot standby instances of the master database that take over as the master database in case of failure.

### **2.5.3. Additional Configuration**

## Load Balancer

Anypoint Platform Private Cloud Edition must be run in a production environment with multiple servers. To distribute traffic among servers and to restrict access to specific ports, you must install and configure a load balance

You can configure your load balancer to use any method for distributing client requests, but in most contexts a round robin strategy is ideal. This load balancer should be reachable through an IP address that is accessible to all machines in your network.

Your load balancer must route the following TCP ports:

In each case, your load balancer must listen on the Load Balancer Port and redirect incoming requests to the Instance Port. Your Anypoint Platform installation includes an internal NGINX server that listens on each of the configured instance ports, then performs the action listed in the Internal Usage column.

Your load balancer should poll the address HTTP:10248/healthz to run a health check on your platform servers and confirm that they are accessible.

## Configure Backup and Restore

After installing Anypoint Platform Private Cloud Edition, you must ensure that you have configured your system to perform regular backups. You must schedule your backup to run at least once per day. However, MuleSoft recommends that your backup runs at least once per hour

1. **Provide Platform IP or Domain**

You need to provide the DNS or IP with which your instance of the Anypoint Platform is exposed to the local network, this is necessary for Mule runtimes to be able to communicate with it. You can easily do this by accessing the **Access Management** section.

## Configure Log Forwarding

Anypoint Platform uses rsyslog to handle logging. To forward these logs to remote hosts you must configure this in the Ops Center.

It does not support to send data to a cloud-based log, so you must forward logs to a 3rd-party log solution like Splunk

## Set up SMTP

In order to handle e-mail alerts for both Runtime Manager and API Manager, you must have an SMTP server configured on your network. You must direct your local installation of Anypoint Platform to this server from the Access Management panel

## SSL Certificate

In order to use Anypoint Platform, you must provide SSL credentials. You can upload a certificate through the Anypoint Platform UI. This certificate must be trusted by every machine that’s connected to the platform

## Use Third Party Authentication

You can have your users log in to Anypoint Platform via various third party authentication tools. You can use LDAP, ping federate, SAML or OpenAM. On this version of the platform, all of your users must be registered in the platform via external identity (except the admin you register via the installation wizard).

## Add a Custom Disclaimer Message

 You can add a custom message to your login page, that all users in your organization will see every time you log in.

### **2.5.4. Pros and Cons**

Pros:

-        Control Plane and Runtime plane are secured by the organization

-        No flow of metadata to the Mulesoft hosted cloud

-        All the exchange assets are strictly remain on premise

Cons

-        Requires expertise to manage the control plane

-        Sometimes its difficult to resolve issues

**Pivotal Cloud Foundry (PCF)** What is Cloud Foundry ?  
  
 Cloud Foundry is an **open source cloud computing platform** originally developed in-house at VMware. It is now owned by Pivotal Software  
  
 What is Pivotal Cloud Foundry (PCF) ?

PCF is a cloud native platform for **deploying and operating modern applications**.

                            PCF can be deployed **on-premises** and on **many cloud providers** to give enterprises a hybrid and multi-cloud platform.

It provides a uniform way for you to launch, and quickly iterate on ideas in the language of your choice.

* **Pivotal Cloud Foundry (PCF)** App 1 
  Build Pack 
  App2 Apps 
  Cm tainer e rwir• 
  , Docker , 
  Infrastructure 

Pivotal cloud foundary hosting a control plane and Runtime plane

Pivotal cloud foundary can be hosted on customer Data center or AWS or GCP or on any cloud providers

If we have an infrastuctutre On top of ‘Infrastructure’ ,PCF provides own  container engine which is  like Garden or Docker or Kubernetes

On top of container engine ,we  deploy our containerized applications.

PCF determines mule applications with the help of ‘Build Pack’

What usaually happens is when ever you create a code ,the code has some thing called as build pack

Job of ‘Build Pack’ is to determine what kind of application it is , mulesoft has created its PCF build packs , using those build packs PCF determines those are mule applications and they would require a mule runtime and accordingly it will help to deploy the application.

Like PCE ,Control Plane and Runtime Plane are hosted in customer’s data center

Unlike manual runtime installation in PCE, runtimes are provisioned automatically using PCF technology .

Unlike PCE, every application will get separate runtime in PCF

* **Buildpacks:**

Buildpacks provide framework and runtime support for apps.

Buildpacks typically examine your apps to determine what dependencies to download and how to configure the apps to communicate with bound services.

* When you push an app, Cloud Foundry automatically detects an appropriate buildpack for it. This buildpack is used to compile or prepare your app for launch.  
    
      
    
   Anypoint Platform for Pivotal Cloud Foundry  
    
   The Anypoint Platform for Pivotal Cloud Foundry (PCF) component is a fully integrated solution  
    
   Like PCE ,Control Plane and Runtime Plane are hosted in customer’s data center  
    
   Unlike manual runtime installation in PCE, runtimes are provisioned automatically using PCF Buildpacks technology .  
    
   Unlike PCE, every application will get separate runtime in PCF

Deployment mode with

Customer hosted Runtime Plane

Customer hosted Control Plane

Control plane installed on Pivotal Cloud Foundry

* Anypoint Platform for PCF uses the following services:

**Runtime Manager Service**: exposes PCF as a deployment target within the Anypoint Runtime Manager and lets you deploy Mule applications to PCF, alongside other deployment targets.

**API Gateway Service**: enables you to create and deploy a Mule API proxy that intercepts traffic to a non-Mule application running on PCF.

 MuleSoft interface to apply policies over the API and

 the PCF interface to view usage and policy metrics.

**Anypoint Metering Service**: ---provides an API

that enables you to view usage metrics for Mule applications deployed to PCF.

 You can view usage metrics for

production,

pre-production,

staging, and

development environments.

   
  
    
  
 **The Anypoint Platform for PCF solution enables you to:**

Deploy Mule applications to PCF------ using the Runtime Manager UI.

Create and deploy a Mule proxy application to PCF. A proxy application intercepts traffic to a non-Mule application, enabling you to apply policies and view usage metrics.

If you are using Anypoint Platform for PCF and you have multiple PCF installations, each must be linked to its own separate installation of Anypoint Platform Private Cloud Edition.

If your Anypoint Platform organization uses multiple environments, it is recommended that you have one PCF space for each.

If you have a PCF installation with multiple spaces, you must run the **Space Configuration Tool** on each of these spaces.

* **Anypoint Platform for PCF uses the following services**

**Runtime Manager Service**: exposes PCF as a deployment target within the Anypoint Runtime Manager and lets you deploy Mule applications to PCF, alongside other deployment targets.

                            Deploy Mule applications to PCF------ using the Runtime Manager UI.

**API Gateway Service**: enables you to create and deploy a Mule API proxy that intercepts traffic to a non-Mule application running on PCF.

                        MuleSoft interface to apply policies over the API and  
  
                                    The PCF interface to view usage and policy metrics.

**Anypoint Metering Service**: ---provides an API

                that enables you to view usage metrics for Mule applications deployed to PCF.  
  
    
  
 **High level Install and configuration steps:**

Install and Configure Pivotal Cloud Foundry (PCF)

Install the Any point platform Service Brokers for PCF tile.

Install the Mule Runtime Build pack

Run the PCF Space Configuration Tool . It installs Any point platform services in a PCF space. We must run this tool for each space in our PCF installation. PCF uses spaces to create different environments for development, staging, and production. The Space Configuration Tool creates API Gateway Service, Runtime Manager Service and PCF Metering Service

(Optional) Configure a Hazelcast Server . A Hazelcast server is required only if we are applying policies or scaling our APIs in a clustered PCF environment.

Deploy Mule applications to PCF from Any point Runtime manager

Access usage data using the PCF Metering API. After our Mule applications are running, we can monitor our PCF usage metrics via the PCF Metering API .

Reference documentation for setup

* **Support:** High Availability: Customer need to configure  
    
   Load Balancing: Customer need to configure  
    
   Logging: No , need to leverage third party logging services like Splunk  
    
   Monitoring: We can only trigger Alerts , can’t see utilization info like in Cloud Hub control plane  
    
   Object Store: Customer need to configure  
    
   Scheduling: No  
    
   Security Updates: Manual  
    
   Shared Resources: Yes  
    
      
    
   **Uses Cases:**

Latency: when your core systems are in one data center and you want your runtime to be in same data center

Compliance requirements: Government specific projects

GDPR and Data residency requirements

## Runtime Behavior / thread pool

**Mule 3 Runtime Architecture**

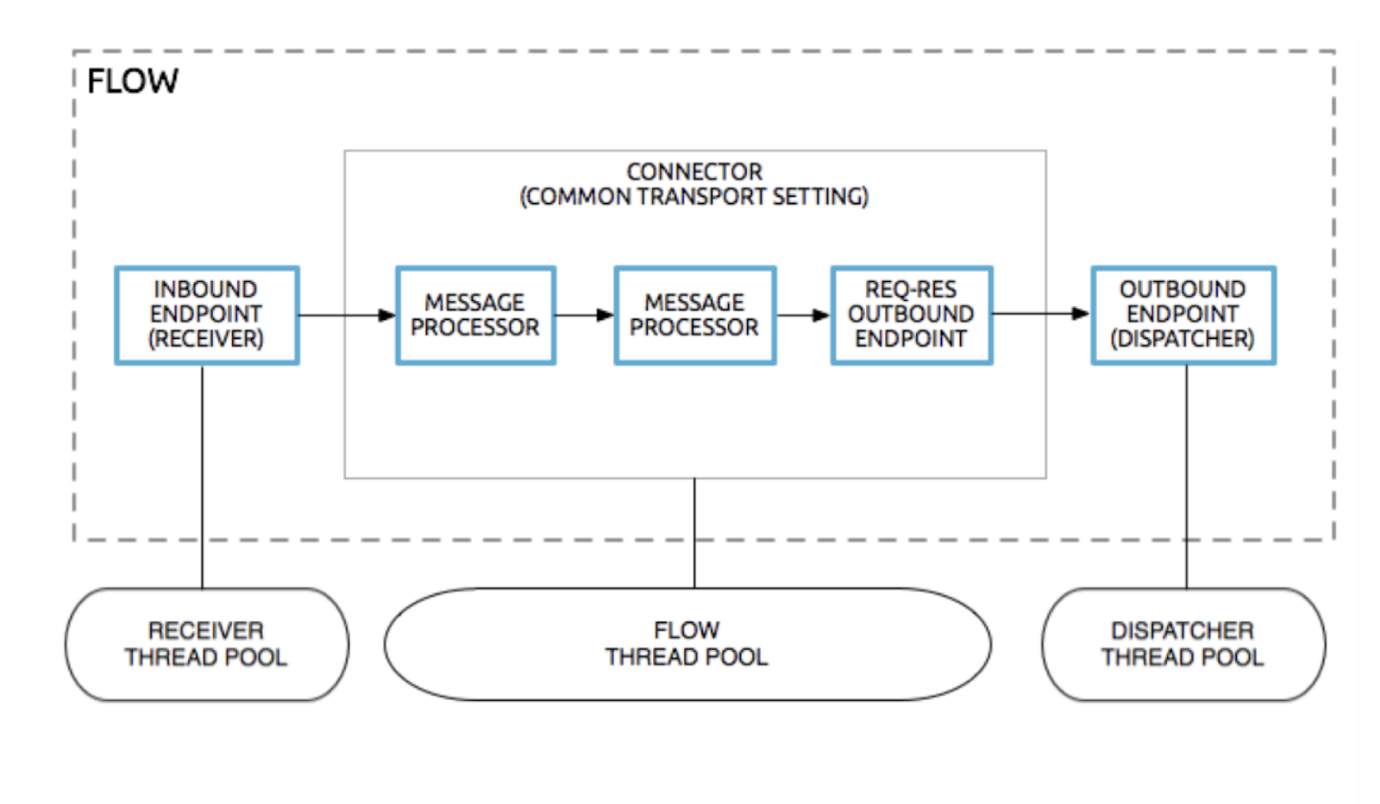
The Architecture upon which Mule 3 was built is SEDA Architecture. SEDA (Staged Event Driven Architecture) is designed to enable high concurrency of services. SEDA decomposes an application into network of stages separated by dynamic resource controllers to allow applications to adjust dynamically to changing load.

Decouple an application into 3 phases if the processing strategy is Asynchronous.

**Receiving**:  Message source’s thread

**Processing**: Message processor’s threads

**Dispatching**: Outbound endpoint’s thread



A thread pool is a collection of available threads. There is a separate thread pool for each receiver, flow and dispatcher.

If you are doing asynchronous processing, the receiver thread is used only to place the message on a staged event-driven architecture (SEDA) queue, at which point the message is transferred to a flow thread, and the receiver thread is released back into the receiver thread pool so it can carry another message. As the message is processed, if it needs to be sent to an outbound endpoint, one of the following applies:

o   If the outbound endpoint is one-way, the message is copied and the copy processed by a dispatcher thread, while the flow thread continues processing the original message in parallel.

o   If the outbound endpoint is request-response, the flow thread sends the message to the outbound endpoint and waits for the response. When the response arrives, the flow threads resume by processing the response.

Several kinds of threads are used to process a message:

* The receiver thread, which originally receives the message, either:
* Processes the entire flow (synchronous), or
* Ends by writing the message to a SEDA queue (asynchronous)
* The flow thread, which processes the bulk of the flow (asynchronous)
* Dispatcher threads, which send messages to one-way endpoints (asynchronous)

## Advantage

## To split the flow into different phases and  fine tune them individually

## Allocate more resources to each phase

## Allocate more threads for each phase

## Pools, queues, and behaviour of this strategy is configurable.

## 

## Disadvantage

## Developers will spend a lot of time in analysing and configuring threads in the thread pool.

## Mulesoft does not want this flexibility of configuring threads to go into Developer’s hand

**Mule 4 Runtime - Reactive Programming**

Why do we need reactive Programming?

Because application requirements have changed dramatically in recent years. Today users expect millisecond response times and 100% uptime.

Today’s demands are simply not met by yesterday’s software architecture.

We want systems that are Responsive, Resilent, Elastic and Message Driven. These systems are called Reactive Systems

Features:

* Responsive:  The system responds in a timely manner if at all possible. Responsive systems focus on providing rapid and consistent response times.
* Resilient: The system stays responsive in the face of failure. This applies not only to highly-available, mission critical systems. Any system that is not resilient will be unresponsive after a failure. Resilient is achieved by containment, isolation and replication.

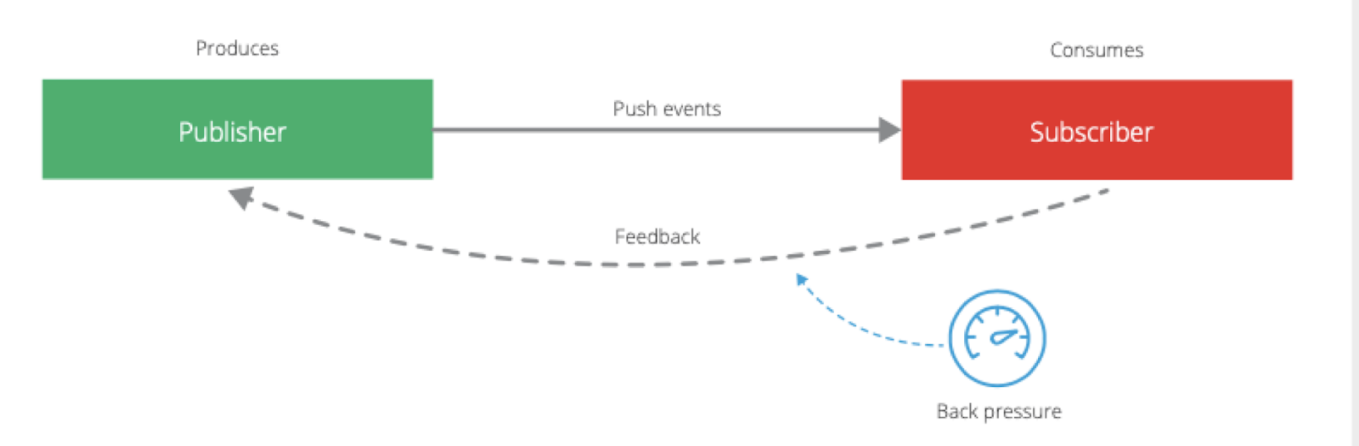
* Elastic: The system stays responsive under varying workload and bottleneck. Reactive Systems can react to changes in the input rate by increasing or decreasing the resources allocated to service these inputs

* Message Driven: Reactive Systems rely on [asynchronous](https://www.reactivemanifesto.org/glossary#Asynchronous) [message-passing](https://www.reactivemanifesto.org/glossary#Message-Driven) to establish a boundary between components that ensures loose coupling, isolation and location transparency.

Mule 4 uses reactive programming to facilitate non-blocking execution of the event processors. This has a significant impact on a Mule application’s ability to scale the amount of events it can handle concurrently.

**Backpressure**

In order to allow the subscriber to consume the event stream without being overwhelmed by too many events, the subscriber must be able to apply back pressure, in other words signal to the producer to “please slow down.”



**Automatic back pressure in Mule 4**

Mule 4 applications are automatically configured so that the event source receives a back pressure signal when all threads are currently executing and no free threads remain in a required thread pool. In practical terms this will trigger the HTTP Listener, for example, to respond with a 503–“Server busy”, and the JMS Listener will not acknowledge receipt of a message. OutOfMemory errors are avoided as a result of this configuration.

**Manual back pressure in Mule 4**

Mule developers can also configure each event processor to signal back pressure to the event source through the “maxConcurrency” attribute. This configuration affects the number of events that can pass through the event processor per second.

**Thread Management and Auto Tuning in Mule 4**

For each event processor in Mule is categorized into different Processing Types

* CPU\_LITE

o   For tasks that take up to 10ms to execute.

o   No blocking IO operations should be executed here

o   Default size is 2 \* cores

o   Logger, HTTP Requester

* CPU\_INTENSIVE
  + For tasks that take more than 10ms to execute (duration is not enforced, but misclassifying tasks has bad consequences).
  + Typically for transformations, encrypt/decrypt, heavy computation, etc.
  + Non-blocking IO operations should be executed here.
  + Default size is 2 \* cores.
  + Dataweave, Scripting

* BLOCKING\_IO
  + All blocking IO operations should happen here.
  + Significantly larger than the other pools, as most threads here are expected to be in a blocked state.
  + Default size comes from a formula that considers the available memory, the default size of the streaming buffers and other concepts.
  + Database, IO

Thread pools are no longer configurable at the level of a Mule application. We now have three centralized pools in Mule 4.1 and 4.2 based on processing types

CPU\_LITE

CPU\_INTENSIVE

BLOCKING\_IO

All three are managed by the Mule runtime and shared across all applications deployed to that runtime. A running Mule application will pull threads from each of those pools as events pass through its processors. The consequence of this is that a single flow may run in multiple threads. Mule 4 optimizes the execution of a flow to avoid unnecessary thread switches.

**Proactor pattern**:

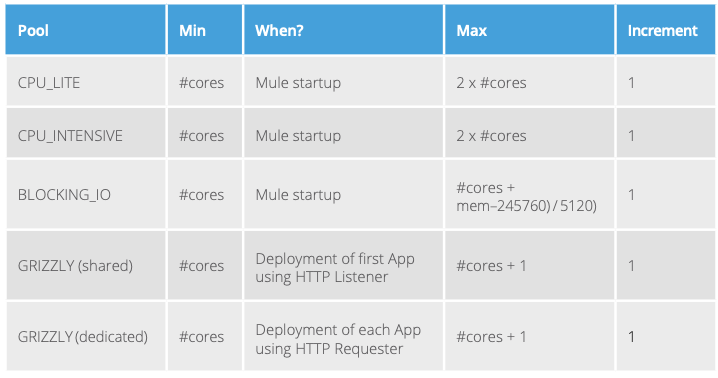
Proactor is a [software design pattern](https://en.wikipedia.org/wiki/Design_pattern_(computer_science)) for [event handling](https://en.wikipedia.org/wiki/Event_handling) in which long running activities are running in an asynchronous part. A completion handler is called after the asynchronous part has terminated. The proactor pattern can be considered to be an [asynchronous](https://en.wiktionary.org/wiki/asynchronous) variant of the [synchronous](https://en.wikipedia.org/wiki/Synchronization_(computer_science)) [reactor pattern](https://en.wikipedia.org/wiki/Reactor_pattern)

**Custom Thread Pool**

Apart from default **THREE pools** Mule 4 runtime use some additional pools for specific purposes:

1. **NIO Selector:**Based on requirement components use NON-Blocking IO. Internally Java [NIO Selector](https://www.javatpoint.com/java-nio-selector) is using most of the time by the connector or component.
2. **Recurring Pools:**Some connectors or components can create this type of custom pool for **recurring tasks**.

* [**GRIZZLY**](https://javaee.github.io/grizzly/)
  + In Mule 4 runtime this one of the most custom thread pools being used **HTTP components**.
  + This is an NIO Selector thread pool. Java [**“NIO”**](https://docs.oracle.com/javase/7/docs/api/java/nio/channels/Selector.html) has the concept of a selector thread.
  + This pool is also configured at **runtime level** and shared by application deployed to that runtime.
  + GRIZZLY is divided into **TWO pools**. One is **GRIZZLY(Shared)** and the other is **GRIZZLY(Dedicated)**. The Shared one will use by **HTTP Listener** and **HTTP Requestor** will use the dedicated one.

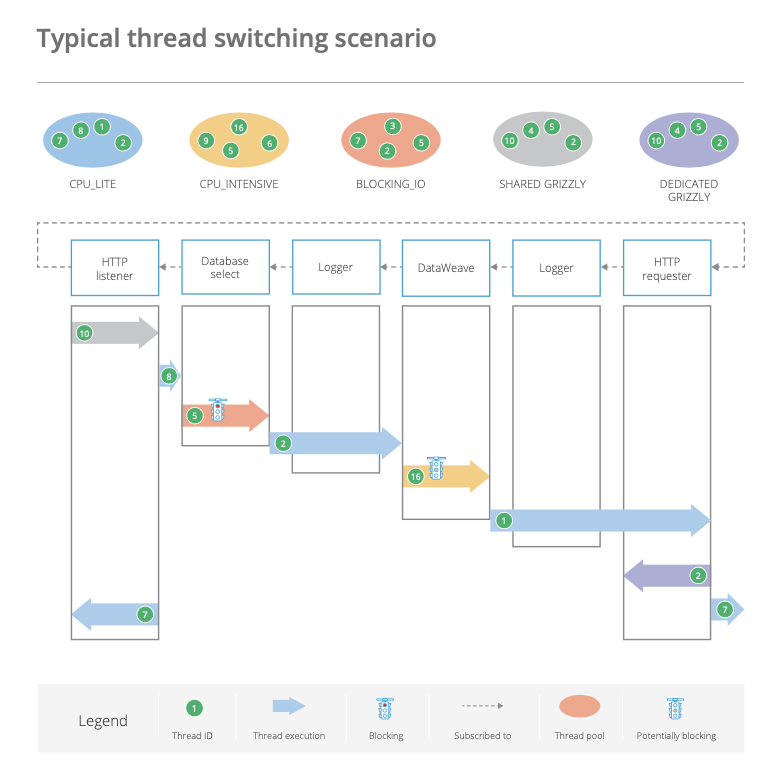


**Mule 4.3- Uber Threads**

Performance is a major topic of the Mule 4.3 release. As part of that, we improved the above model by unifying these three thread pools into a single thread pool. This helps us improve the Mule runtime’s auto-tuning feature and make better use of available resources.

It is called UBER POOL

We can configure pooling strategy in scheduler-conf.properties file



## JVM properties

If we want to pass additional arguments to the JVM that runs Mule, we can add wrapper.java.additional.n entries to the wrapper.conf file in ${MULE\_HOME}/conf  directory or Use the -M command line argument

**JVM Tuning**

**JVM tuning** mainly involves optimizing the garbage collector for better collection performance so that applications running on VMs can have a larger throughput while using less memory and experiencing lower latency.

We can modify Heap Memory and MetaSpace Memory for better performance

Heap size : Allocate more memory when we have to run large payload or when multiple programs are running in a runtime

**MetaSpaceSize**:  Metaspace is memory the VM uses to store class metadata. Class metadata are the runtime representation of java classes within a JVM process - basically any information the JVM needs to work with a Java class. That includes, but is not limited to, runtime representation of data from the JVM class file format. Allocate more memory for internal processing. Incase to load more class or class loaders.

**Garbage collection** (GC) in the Java virtual machine (JVM) to automatically determine what memory is no longer being used by a Java application and to recycle this memory for other uses.

**Parallel Garbage collector** (also known as throughput collector) is a garbage collector which uses multiple threads to speed up garbage collection.

The Parallel collector is enabled with the command-line-option -XX:+UseParallelGC

Its the by default option

This is used where performance time is crucial, throughput is crucial

**Concurrent Mark and Sweep (CMS)** collector is designed for applications that prefershorter garbage collection pauses and that can afford to share processor resources with the garbage collector while the application is running

The CMS collector is enabled with the command-line option -XX:+UseConcMarkSweepGC.

This is used when response time is crucial

## 