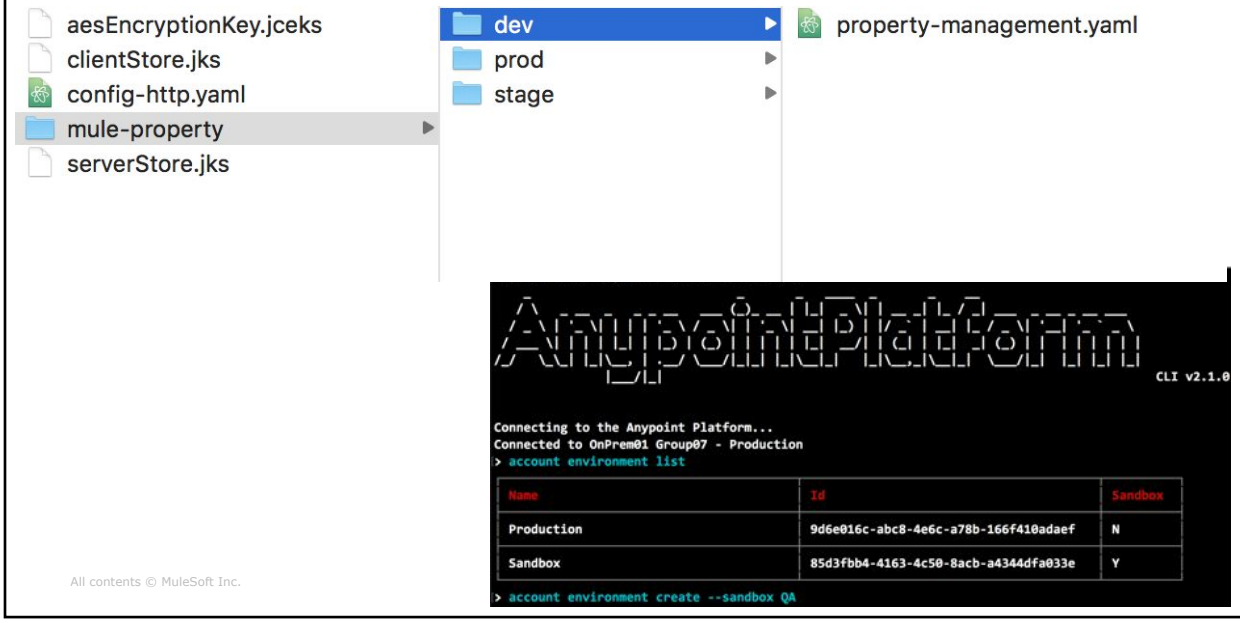
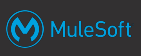
# Module 10: Designing an Efficient and Automated Software Development Lifecycle

## Goal



## At the end of this module, you should be able to



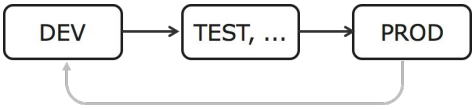

- Manage property files for Mule applications across different environments
- Manage Anypoint Platform environments for Mule application deployments
- Design testing strategies for Mule applications
- Implement continuous integration and continuous delivery (CI/CD) for an organization
- Automate deployment and management with Anypoint Platform




# Managing property files for different environments

## Using properties to help Mule applications evolve

- Mule applications may need to use different configurations between Mule runtimes
  - Examples
    - To avoid TCP bind errors
    - To configure values for a SDLC environment
    - To configure values for a region
  - Runtime Manager provides a UI to configure Mule application properties
  - Mule application properties can also be configured without Runtime Manager



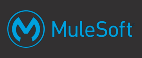

- These phases/regions/environments have differences
  - They use different data (Live vs. non-live data)
    - Different databases or other systems of record, with different restrictions
  - They often use different credentials to access data
    - Different values
    - Non-encrypted vs encrypted




## Configuring Mule application properties

- Properties that might change include
  - Connector properties
    - Location information like hostnames and ports
    - Performance tuning values like connection pools, thread pools, and timeout values
    - Security properties
    - Names of environments, files, and resources
    - External locations for files and other resources
  - Performance tuning properties
    - Connection properties such as reconnection limits and connection pools
    - Other tuning properties like time-out values and polling frequencies
  - Other application properties like JMS message configurations

## Managing Mule application and system level properties using a Mule Runtime

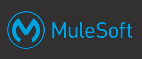


- Mule application level properties
  - Only visible to the deployed Mule application
  - Stored in the Mule application deployable JAR
- System level properties
  - Set as JVM system variables
  - Shared by every Mule application deployed into the Mule runtime
  - Can be set in the wrapper.conf file for from the command line when starting the Mule runtime
  - Will override Mule application level properties set by the Mule appliction's deployable JAR

https://support.mulesoft.com/s/article/How-can-I-set-Mule-and-Java-system-properties-at-startup




## Architects need to design a way to externalize Mule application properties



- Configuration values that change should be externalized by developers into configuration files
  - They should not be hard-coded inside Mule applications
- External configuration files make Mule applications easier to upgrade and migrate
- External configuration management systems can also be used, but require more work to set up and manage
  - Typically requires development of a custom integration with the Mule runtime

## Separating configurations by environment

- Create a separate configuration file for some/all environment

```
dev-prop.yaml
sqldb:
  user: "mule"
  password: "mulemax"
```

```
prod-prop.yaml
sqldb:
  user: "mule"
  password: "![gCs37bbR6NDrTrABIxHhOA==]"
```

- Details in the Anypoint Platform Operations: Customer-Hosted Runtimes course




# Managing environments across an organization

## Organization owner

- The **Organization Owner** is the Anypoint Platform user that first signs up for an Anypoint Platform account
  - It is the user that pays for the Anypoint Platform account
  - This is not a role but an identifier for this single user
  - Inherits the **Organization Administrator role** by **default**
- This upper level organization has a client id and client secret used to secure some Anypoint Platform features
- The upper level organization has either one external idP for identity management or none
  - In which case Anypoint Platform performs identify management




## Managing business groups, users, roles, and environments

- Each organization can contain child **business groups**
- The organization and its business groups can each define different roles, environments, and users
- Roles define and manage user permissions across Anypoint Platform, for specific environments within a particular business group
  - Some predefined roles are created for every business group
  - Custom roles can also collect a different set of permissions

## More details about using business groups

- Business groups provide complete **isolation** of resources
- vCores are **assigned** to a specific business groups
  - Makes those vCores only available to the business group and unavailable to the parent organization or business group
- Each business group has its **own** environments
- Each business group has a **separate** client id and client secret
- Deleting a business group is NOT recoverable as all resources get deleted




## How to delegate administration of an organization and its child business groups

- The **organization owner** has **all permissions** for its organization and child business groups, independent of any roles
- Other users can be added to the Administrator role of the upper organization, and then also have **all permissions**
- Each **business group** also has an **owner** with full administrator role privileges to that business group and its child business groups
  - Has full permission to create, change, or remove any role, user, or child business group
  - But **not** to **parent** business groups

## Managing environments for deployments

- Anypoint Platform supports the following **types** of environments
  - **Production** quality environments
    - Where you can deploy Mule applications and APIs publicly
    - When you create a new Anypoint Platform account, by default it contains one production environment
  - **Sandbox** quality environments
    - Provide useful environments for development and testing
    - By default, Anypoint Platform accounts are created with one sandbox environment
  - **Design** quality environments
    - Enables you to test and run Mule applications at design time
    - By default, Anypoint Platform accounts are created with one design environment



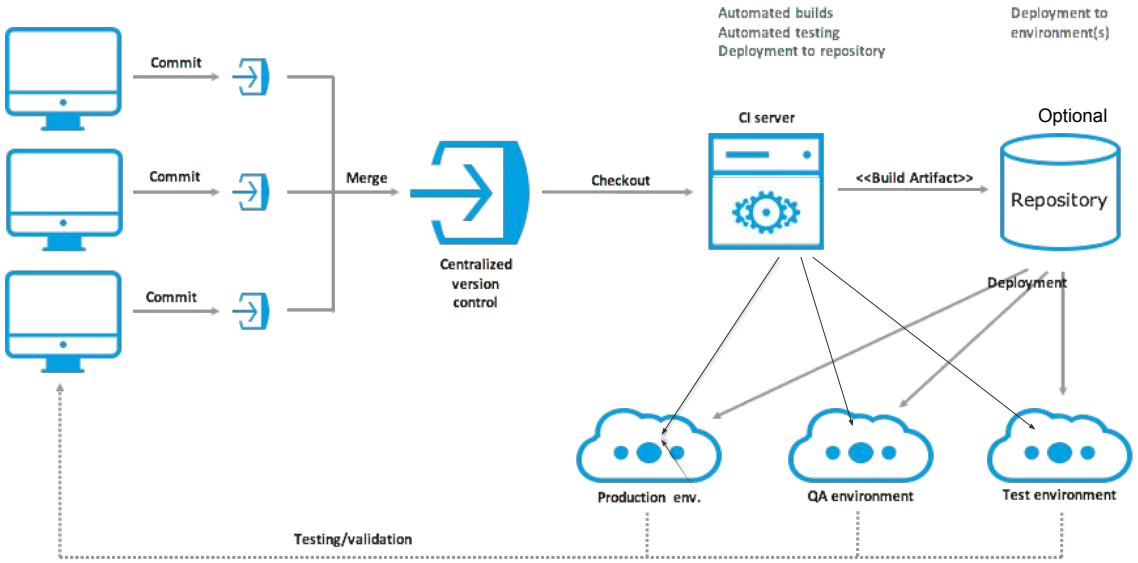

## Managing VPC in environments

- Anypoint Platform licenses the number of VPCs available to an organization
- An Anypoint Platform VPC can be configured to span across multiple environments and child business groups under the current business group
- Each VPC is assigned to a business group

Reference: Anypoint Platform Operations: CloudHub

# Implementing continuous integration and continuous delivery (CI/CD)

## Automation using CI CD

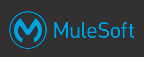


## Implementing CI CD on Anypoint platform


- Anypoint Platform supports continuous integration and continuous delivery using industry standard tools
  - The **Mule Maven plugin** can automate building, packaging, and deployment of Mule applications from source projects
  - The **MUnit Maven plugin** can automate test execution, and ties in with the Mule Maven plugin




## Implementing CI CD on Anypoint platform using Anypoint Exchange




- **Anypoint Exchange** is a central repository for various types of assets
  - So they can be discovered and reused to build your integration projects
- Stores and shares enterprise assets
  - RAML fragments
  - Custom Connectors
  - Project templates and examples
- Is a Maven compatible artifact repository
  - Is not intended as a replacement for a full version control system

## Using Maven with Mule 4 applications

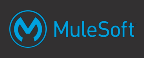
- Anypoint Studio includes Maven support
- Anypoint Studio can generate and manage the Mule application's pom.xml file automatically
  - For example, adding a Salesforce connector imports a compatible SFDC library
- All Anypoint Studio Mule projects are automatically **mavenized**
- The Maven plugin supports Mule domain and Mule application projects




## How the Maven plugins supports Mule application lifecycle phases

- Together, the Mule and MUnit Maven plugins implement MuleSoft behavior in various Maven lifecycle phases
  - compile - Compile the Mule source code of the project
  - test - Runs MUnit tests associated with project
  - package - Packages the project into a Mule deployable jar
  - install - Sends the distributable to $MULE_HOME, a local repository
  - deploy – Sends the distributable to a remote repository
- Triggering a later phase in the lifecycle triggers all phases before it

## Deploying a Maven build artifact to a Mule runtime



- To use Maven to deploy an application to a server, use the mule-maven-plugin
- Supports deployments to
  - CloudHub
  - Anypoint Runtime Manager
  - Standalone runtime
  - Mule Agent
- For more information on deployment for specific scenario
  - https://docs.mulesoft.com/mule-runtime/4.1/mmp-deployment-concept




## How to implement shared resources in Mule project

- Every Mule application can be assigned to one Mule domain in the Mule application's configurations pom.xml file
- The Mule domain project is referenced in each of the Mule applications that are meant to use these share resources
- Mule applications can be associated with **only one** domain at a time
- Maven supports creation of Mule domain project

## Limitations of domain projects

- Mule domain projects can only be manually deployed to customer-hosted Mule runtimes
  - Runtime Manager **does not support** deployment or management of Mule domains, but you can deploy and manage Mule applications to an existing Mule domain already running in the Mule runtime(s)
- Runtime Manager does not support deployment of domain project




## Resource to understand how to design and implement CI/CD for Mule applications

- The student files have documents to help with CI/CD
- Reference: Anypoint Platform Development: Advanced

## Reflection questions

MuleSoft

- When are Mule domains useful, and what are the tradeoffs of coupling Mule applications with a Mule domain?
- Why are Mule domains not needed in CloudHub?
- When and why would  Mule domains be useful in Runtime Fabric deployments?
- How are these shared resources implemented and how do they behave between Mule applications in a Mule domain?
  - HTTP Listener
  - HTTP Request
  - VM Listener
  - Object Store connector
  - File Listener (On New or Updated File operation)
  - Database Listener (On Table Row operation)




# Automating Anypoint Platform

## Automation on Anypoint platform

- Anypoint Platform and the Mule Maven plugin both provide multiple option for automation
  - Anypoint Command Line Interface (CLI)
  - Anypoint Platform APIs
- Automate Anypoint Platform administrative activities with the
  - Anypoint-CLI command-line tool
    - Combines REST API steps into easier to use commands
  - Anypoint Platform APIs
    - First generate an access token or use OAuth2
    - Use the access token in other REST calls
    - Set the environment id or organization id as needed




## Automation using Anypoint Command Line Interface (CLI)

- Is a Node.js based application
- Used to access Anypoint Platform APIs
- Anypoint CLI commands simplify common use cases
  - Authentication using using username/password not secure access token
  - Set organization, environment using name rather than the ids
  - Can look up resources using name rather than id
- Runs in interactive or non-interactive mode
  - In interactive mode, the user type into command line interface to perform automation of tasks
  - In non-interactive mode, the user create script file to perform automation of tasks and is preferred for repetitive tasks
- Reference
  - https://docs.mulesoft.com/runtime-manager/anypoint-platform-cli

## Automation using Anypoint platform APIs

- Anypoint Platform APIs than can help orchestrate API based deployment and management of CI/CD automation
  - MuleSoft Developer Portal
    - https://anypoint.mulesoft.com/exchange/portals/anypoint-platform/
  - Access Management API
    - https://anypoint.mulesoft.com/exchange/portals/anypoint-platform/f1e97bc6-315a-4490-82a7-23abe036327a.anypoint-platform/access-management-api/
  - CloudHub API
    - The CloudHub Public API enables you to access application management services for applications deployed to CloudHub
    - https://anypoint.mulesoft.com/exchange/portals/anypoint-platform/f1e97bc6-315a-4490-82a7-23abe036327a.anypoint-platform/cloudhub-api/




## Automation using Anypoint platform APIs (cont.)

- Other Anypoint Platform APIs than can help orchestrate API based deployment and management of CI/CD automation
  - API Manager API
    - The API Manager API enables you to manage an API by applying policies, setting SLAs, configuring alerts for your API instances, and promoting API instances
    - https://anypoint.mulesoft.com/exchange/portals/anypoint-platform/f1e97bc6-315a-4490-82a7-23abe036327a.anypoint-platform/api-manager-api/
  - API platform v2
    - The API Platform API exposes the management capabilities of the Anypoint Platform for APIs, enabling them to be used by external sites
    - https://anypoint.mulesoft.com/exchange/portals/anypoint-platform/f1e97bc6-315a-4490-82a7-23abe036327a.anypoint-platform/api-platform-api/
  - Anypoint Exchange API and Proxies API
    - Please refer https://docs.mulesoft.com/release-notes/upgrade

## Reflection questions

- When and how would you use the Anypoint CLI?
- When would you use the Anypoint Platform REST APIs instead, or in conjunction with the Anypoint CLI?
- What are some ways you can build automated scripts with Anypoint CLI and the Anypoint Platform REST APIs?




# Summary

# Summary

- DevOps requires having a strategy to manage properties for different environments
- MUnit promotes test driven development for an organization
- Simplifying CI and CD pipelines is also key for organizational success
- MUnit helps developers to unit test APIs throughout a SDLC