# Module : [Apex Triggers](https://trailhead.salesforce.com/content/learn/modules/apex_triggers?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

## [Get Started with Apex Triggers](https://trailhead.salesforce.com/content/learn/modules/apex_triggers/apex_triggers_intro?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

AccountAddressTrigger.apxt

//code

trigger AccountAddressTrigger on Account (before insert, before update) {

for(Account account:Trigger.New){ if(account.Match\_Billing\_Address c == True){

account.ShippingPostalCode = account.BillingPostalCode;

}

}

}

[**Bulk Apex Triggers**](https://trailhead.salesforce.com/content/learn/modules/apex_triggers/apex_triggers_bulk?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

ClosedOpportunityTrigger.apxt

//code

trigger ClosedOpportunityTrigger on Opportunity (after insert, after update) {

List<task> tasklist = new List<Task>();

for(Opportunity opp : Trigger.New){ if(opp.StageName == 'Closed Won'){

tasklist.add(new Task(subject = 'Follow Up Test Task', WhatId = opp.Id));

}

}

if(tasklist.size()>0){ insert tasklist;

}

}

# Module : [Apex Testing](https://trailhead.salesforce.com/content/learn/modules/apex_testing?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

## [Get Started with Apex Unit Tests](https://trailhead.salesforce.com/content/learn/modules/apex_testing/apex_testing_intro?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

TestVerifyDate.apxc

//code @isTest

public class TestVerifyDate {

@isTest static void testOldDate(){

Date dateTest = VerifyDate.CheckDates(date.today(), date.today().addDays(-1));

System.assertEquals(date.newInstance(2022, 5, 30), dateTest);

}

@isTest static void testLessThan30Days(){

Date dateTest = VerifyDate.CheckDates(date.today(), date.today().addDays(20));

System.assertEquals(date.today().addDays(20), dateTest);

}

@isTest static void testMoreThan30Days(){

Date dateTest = VerifyDate.CheckDates(date.today(), date.today().addDays(31));

System.assertEquals(date.newInstance(2022, 5, 30), dateTest);

}

}

## [Test Apex Triggers](https://trailhead.salesforce.com/content/learn/modules/apex_testing/apex_testing_triggers?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

TestRestrictContactByName.apxc

//code @isTest

private class TestRestrictContactByName { @isTest static void testInvalidName() {

Contact myConact = new Contact(LastName='INVALIDNAME');

insert myConact; Test.startTest();

Database.SaveResult result = Database.insert(myConact,

false);

Test.stopTest(); System.assert(!result.isSuccess()); System.assert(result.getErrors().size() > 0); System.assertEquals('Cannot create contact with

invalid last name.',

));

}

}

## [Create Test Data for Apex Tests](https://trailhead.salesforce.com/content/learn/modules/apex_testing/apex_testing_data?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

result.getErrors()[0].getMessage(

RandomContactFactory.apxc

//code

public class RandomContactFactory {

public static List<Contact> generateRandomContacts(Integer numcnt, string lastname){

List<Contact> contacts = new List<Contact>(); for(Integer i=0;i<numcnt;i++){

contact cnt = new Contact(FirstName = 'Test '+ i, LastName = lastname);

contacts.add(cnt);

}

return contacts;

}

}

# Module : [Asynchronous Apex](https://trailhead.salesforce.com/content/learn/modules/asynchronous_apex?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

## [Use Future Methods](https://trailhead.salesforce.com/content/learn/modules/asynchronous_apex/async_apex_future_methods?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

AccountProcessor.apxc

//code

public class AccountProcessor { @future

public static void countContacts(List<Id> accountIds){

List<Account> accountsToUpdate = new List<Account>();

List<Account> accounts = [Select Id, Name, (Select Id from Contacts) from Account where Id in :accountIds];

for(Account acc:accounts){

List<Contact> contactList = acc.Contacts; acc.Number\_Of\_Contacts c = contactList.size(); accountsToUpdate.add(acc);

}

update accountsToUpdate;

}

}

AccountProcessorTest.apxc

//code @IsTest

private class AccountProcessorTest { @IsTest

private static void testcountContacts(){

Account newAccount = new Account(Name = 'Test Account');

insert newAccount;

contact newContact1 = new Contact(FirstName = 'John', LastName = 'Doe', AccountId = newAccount.Id);

insert newContact1;

contact newContact2 = new Contact(FirstName = 'Jone', LastName = 'Doe', AccountId = newAccount.Id);

insert newContact2;

List<Id> accountIds = new List<Id>(); accountIds.add(newAccount.Id);

Test.startTest(); AccountProcessor.countContacts(accountIds); Test.stopTest();

}

}

## [Use Batch Apex](https://trailhead.salesforce.com/content/learn/modules/asynchronous_apex/async_apex_batch?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

LeadProcessor.apxc

//code

global class LeadProcessor implements Database.Batchable<sObject> {

global Integer count = 0;

global Database.QueryLocator start (Database.BatchableContext bc) {

return Database.getQueryLocator('Select Id, LeadSource from lead');

}

global void execute (Database.BatchableContext bc,List<Lead> l\_lst) {

List<lead> l\_lst\_new = new List<lead>(); for(lead l : l\_lst) {

l.leadsource = 'Dreamforce'; l\_lst\_new.add(l);

count+=1;

}

update l\_lst\_new;

}

global void finish (Database.BatchableContext bc) {

system.debug('count = '+count);

}

}

LeadProcessorTest.apxc

//code @isTest

public class LeadProcessorTest {

@isTest

public static void testit() {

List<lead> l\_lst = new List<lead>(); for (Integer i = 0; i<200; i++) {

Lead l = new lead(); l.LastName = 'name'+i; l.company = 'company'; l.Status = 'somestatus'; l\_lst.add(l);

}

insert l\_lst; test.startTest();

Leadprocessor lp = new Leadprocessor(); Id batchId = Database.executeBatch(lp); Test.stopTest();

}

}

## [Control Processes with Queueable Apex](https://trailhead.salesforce.com/content/learn/modules/asynchronous_apex/async_apex_queueable?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

AddPrimaryContact.apxc

//code

public class AddPrimaryContact implements Queueable { public contact c;

public String state;

public AddPrimaryContact(Contact c, String state) { this.c = c;

this.state = state;

}

public void execute(QueueableContext qc) { system.debug('this.c = '+this.c+' this.state =

'+this.state);

List<Account> acc\_lst = new List<account>([select id, name, BillingState from account where account.BillingState =

:this.state limit 200]);

List<contact> c\_lst = new List<contact>(); for(account a: acc\_lst) {

contact c = new contact();

c = this.c.clone(false, false, false, false); c.AccountId = a.Id;

c\_lst.add(c);

}

insert c\_lst;

}

}

AddPrimaryContactTest.apxc

//code @IsTest

public class AddPrimaryContactTest { @IsTest

public static void testing() {

List<account> acc\_lst = new List<account>(); for (Integer i=0; i<50;i++) {

account a = new account(name=string.valueOf(i),billingstate='NY');

system.debug('account a = '+a); acc\_lst.add(a);

}

for (Integer i=0; i<50;i++) { account a = new

account(name=string.valueOf(50+i),billingstate='CA'); system.debug('account a = '+a); acc\_lst.add(a);

}

insert acc\_lst; Test.startTest();

contact c = new contact(lastname='alex'); AddPrimaryContact apc = new AddPrimaryContact(c,'CA'); system.debug('apc = '+apc);

System.enqueueJob(apc); Test.stopTest();

List<contact> c\_lst = new List<contact>([select id from contact]);

Integer size = c\_lst.size(); system.assertEquals(50, size);

}

}

## [Schedule Jobs Using the Apex Scheduler](https://trailhead.salesforce.com/content/learn/modules/asynchronous_apex/async_apex_scheduled?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

DailyLeadProcessor.apxc

//code

public class DailyLeadProcessor implements schedulable{

public void execute(schedulableContext sc) { List<lead> l\_lst\_new = new List<lead>(); List<lead> l\_lst = new List<lead>([select id,

leadsource from lead where leadsource = null]); for(lead l : l\_lst) {

l.leadsource = 'Dreamforce'; l\_lst\_new.add(l);

}

update l\_lst\_new;

}

}

DailyLeadProcessorTest.apxc

//code @isTest

public class DailyLeadProcessorTest { @testSetup

static void setup(){

List<Lead> lstOfLead = new List<Lead>(); for(Integer i = 1; i <= 200; i++){

Lead ld = new Lead(Company = 'Comp' + i ,LastName

= 'LN'+i, Status = 'Working - Contacted'); lstOfLead.add(ld);

}

Insert lstOfLead;

}

static testmethod void testDailyLeadProcessorScheduledJob(){

String sch = '0 5 12 \* \* ?'; Test.startTest();

String jobId = System.schedule('ScheduledApexTest', sch, new DailyLeadProcessor());

List<Lead> lstOfLead = [SELECT Id FROM Lead WHERE LeadSource = null LIMIT 200];

System.assertEquals(200, lstOfLead.size());

Test.stopTest();

}

}

# Module : [Apex Integration Services](https://trailhead.salesforce.com/content/learn/modules/apex_integration_services?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

## [Apex REST Callouts](https://trailhead.salesforce.com/content/learn/modules/apex_integration_services/apex_integration_rest_callouts?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

AnimalLocator.apxc

//code

public class AnimalLocator{

public static String getAnimalNameById(Integer x){ Http http = new Http();

HttpRequest req = new HttpRequest(); req.setEndpoint('https://th-apex-http-

callout.herokuapp.com/animals/' + x); req.setMethod('GET');

Map<String, Object> animal= new Map<String, Object>(); HttpResponse res = http.send(req);

if (res.getStatusCode() == 200) { Map<String, Object> results = (Map<String,

Object>)JSON.deserializeUntyped(res.getBody());

animal = (Map<String, Object>) results.get('animal');

}

return (String)animal.get('name');

}

}

AnimalLocatorTest.apxc

//code @isTest

private class AnimalLocatorTest{

@isTest static void AnimalLocatorMock1() { Test.setMock(HttpCalloutMock.class, new

AnimalLocatorMock());

string result = AnimalLocator.getAnimalNameById(3); String expectedResult = 'chicken'; System.assertEquals(result,expectedResult );

}

}

AnimalLocatorMock.apxc

//code @isTest

global class AnimalLocatorMock implements HttpCalloutMock { global HTTPResponse respond(HTTPRequest request) {

HttpResponse response = new HttpResponse(); response.setHeader('Content-Type',

'application/json');

response.setBody('{"animals": ["majestic badger", "fluffy bunny", "scary bear", "chicken", "mighty moose"]}');

response.setStatusCode(200); return response;

}

}

## [Apex SOAP Callouts](https://trailhead.salesforce.com/content/learn/modules/apex_integration_services/apex_integration_soap_callouts?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

ParkLocator.apxc

//code

public class ParkLocator {

public static string[] country(string theCountry) { ParkService.ParksImplPort parkSvc =

new ParkService.ParksImplPort(); // remove space return parkSvc.byCountry(theCountry);

}

}

ParkLocatorTest.apxc

//code @isTest

private class ParkLocatorTest {

@isTest static void testCallout() { Test.setMock(WebServiceMock.class, new ParkServiceMock

());

String country = 'United States';

List<String> result = ParkLocator.country(country); List<String> parks = new List<String>{'Yellowstone',

'Mackinac National Park', 'Yosemite'}; System.assertEquals(parks, result);

}

}

ParkServiceMock.apxc

//code @isTest

global class ParkServiceMock implements WebServiceMock { global void doInvoke(

Object stub, Object request,

Map<String, Object> response, String endpoint,

String soapAction, String requestName, String responseNS, String responseName, String responseType)

ParkService.byCountryResponse response\_x = new ParkService.byCountryResponse();

response\_x.return\_x = new List<String>{'Yellowstone', 'Mackinac National Park', 'Yosemite'};

response.put('response\_x', response\_x);

}

}

## [Apex Web Services](https://trailhead.salesforce.com/content/learn/modules/apex_integration_services/apex_integration_webservices?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

[AccountManager.apxc](https://trailhead.salesforce.com/content/learn/modules/apex_integration_services/apex_integration_webservices?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

//code @RestResource(urlMapping='/Accounts/\*/contacts') global class AccountManager {

@HttpGet

global static Account getAccount() { RestRequest req = RestContext.request; String accId =

req.requestURI.substringBetween('Accounts/', '/contacts'); Account acc = [SELECT Id, Name, (SELECT Id, Name FROM

Contacts)

FROM Account WHERE Id = :accId];

return acc;

}

}

AccountManagerTest.apxc

//code @isTest

private class AccountManagerTest {

private static testMethod void getAccountTest1() { Id recordId = createTestRecord();

RestRequest request = new RestRequest(); request.requestUri =

'https://na1.salesforce.com/services/apexrest/Accounts/'+ recordId +'/contacts' ;

request.httpMethod = 'GET'; RestContext.request = request;

Account thisAccount = AccountManager.getAccount(); System.assert(thisAccount != null); System.assertEquals('Test record', thisAccount.Name);

}

static Id createTestRecord() { Account TestAcc = new Account(

Name='Test record'); insert TestAcc;

Contact TestCon= new Contact( LastName='Test',

AccountId = TestAcc.id); return TestAcc.Id;

}

}

# Superbadge : [Apex Specialist](https://trailhead.salesforce.com/content/learn/superbadges/superbadge_apex?trailmix_creator_id=trailblazerconnect&trailmix_slug=salesforce-developer-catalyst)

## Automate record creation

MaintenanceRequestHelper.apxc

//code

public with sharing class MaintenanceRequestHelper { public static void updateworkOrders(List<Case>

updWorkOrders, Map<Id,Case> nonUpdCaseMap) { Set<Id> validIds = new Set<Id>(); For (Case c : updWorkOrders){

if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status == 'Closed'){

if (c.Type == 'Repair' || c.Type == 'Routine

Maintenance'){

validIds.add(c.Id);

}

}

}

if (!validIds.isEmpty()){

List<Case> newCases = new List<Case>(); Map<Id,Case> closedCasesM = new

Map<Id,Case>([SELECT Id, Vehicle c, Equipment c, Equipment r.Maintenance\_Cycle c,(SELECT Id,Equipment c,Quantity c FROM Equipment\_Maintenance\_Items r)

Case WHERE Id IN :validIds]);

Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();

AggregateResult[] results = [SELECT Maintenance\_Request c,

MIN(Equipment r.Maintenance\_Cycle c)cycle FROM

FROM

Equipment\_Maintenance\_Item c WHERE Maintenance\_Request c IN

:ValidIds GROUP BY Maintenance\_Request c];

for (AggregateResult ar : results){ maintenanceCycles.put((Id)

ar.get('Maintenance\_Request c'), (Decimal) ar.get('cycle'));

}

for(Case cc : closedCasesM.values()){ Case nc = new Case (

ParentId = cc.Id, Status = 'New',

Subject = 'Routine Maintenance', Type = 'Routine Maintenance', Vehicle c = cc.Vehicle c, Equipment c =cc.Equipment c, Origin = 'Web',

Date\_Reported c = Date.Today()

);

If (maintenanceCycles.containskey(cc.Id)){ nc.Date\_Due c =

Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));

}

newCases.add(nc);

}

insert newCases;

List<Equipment\_Maintenance\_Item c> clonedWPs = new List<Equipment\_Maintenance\_Item c>();

for (Case nc : newCases){

for (Equipment\_Maintenance\_Item c wp : closedCasesM.get(nc.ParentId).Equipment\_Maintenance\_Items r){

Equipment\_Maintenance\_Item c wpClone =

wp.clone();

}

wpClone.Maintenance\_Request c = nc.Id; ClonedWPs.add(wpClone);

}

insert ClonedWPs;

}

}

}

MaitenanceRequest.apxt

//code

trigger MaintenanceRequest on Case (before update, after update) {

if(Trigger.isUpdate && Trigger.isAfter){ MaintenanceRequestHelper.updateWorkOrders(Trigger.New,

Trigger.OldMap);

}

}

## Synchronize Salesforce data with an external system

WarehouseCalloutService.apxc

//code

public with sharing class WarehouseCalloutService {

private static final String WAREHOUSE\_URL = 'https://th- superbadge-apex.herokuapp.com/equipment';

public static void runWarehouseEquipmentSync(){ Http http = new Http();

HttpRequest request = new HttpRequest(); request.setEndpoint(WAREHOUSE\_URL); request.setMethod('GET');

HttpResponse response = http.send(request); List<Product2> warehouseEq = new List<Product2>(); if (response.getStatusCode() == 200){

List<Object> jsonResponse = (List<Object>)JSON.deserializeUntyped(response.getBody());

System.debug(response.getBody()); for (Object eq : jsonResponse){

Map<String,Object> mapJson = (Map<String,Object>)eq;

Product2 myEq = new Product2();

myEq.Replacement\_Part c = (Boolean) mapJson.get('replacement');

myEq.Name = (String) mapJson.get('name'); myEq.Maintenance\_Cycle c = (Integer)

mapJson.get('maintenanceperiod');

myEq.Lifespan\_Months c = (Integer) mapJson.get('lifespan');

myEq.Cost c = (Decimal) mapJson.get('lifespan');

myEq.Warehouse\_SKU c = (String) mapJson.get('sku');

myEq.Current\_Inventory c = (Double) mapJson.get('quantity');

warehouseEq.add(myEq);

}

if (warehouseEq.size() > 0){ upsert warehouseEq;

System.debug('Your equipment was synced with the warehouse one');

System.debug(warehouseEq);

}

}

}

}

## Schedule synchronization

WarehouseSyncShedule.apxc

//code

global class WarehouseSyncSchedule implements Schedulable { global void execute(SchedulableContext ctx) {

WarehouseCalloutService.runWarehouseEquipmentSync();

}

}

## Test automation logic

MaintenanceRequestHelperTest.apxc

//code @istest

public with sharing class MaintenanceRequestHelperTest {

private static final string STATUS\_NEW = 'New'; private static final string WORKING = 'Working'; private static final string CLOSED = 'Closed'; private static final string REPAIR = 'Repair'; private static final string REQUEST\_ORIGIN = 'Web'; private static final string REQUEST\_TYPE = 'Routine

Maintenance';

private static final string REQUEST\_SUBJECT = 'Testing subject';

PRIVATE STATIC Vehicle c createVehicle(){ Vehicle c Vehicle = new Vehicle C(name =

'SuperTruck');

return Vehicle;

}

PRIVATE STATIC Product2 createEq(){ product2 equipment = new product2(name =

'SuperEquipment',

10,

= 10,

true);

}

return equipment;

lifespan\_months C = maintenance\_cycle C replacement\_part c =

PRIVATE STATIC Case createMaintenanceRequest(id vehicleId, id equipmentId){

case cs = new case(Type=REPAIR,

Status=STATUS\_NEW, Origin=REQUEST\_ORIGIN, Subject=REQUEST\_SUBJECT,

Equipment c=equipmentId, Vehicle c=vehicleId);

return cs;

}

PRIVATE STATIC Equipment\_Maintenance\_Item c createWorkPart(id equipmentId,id requestId){

Equipment\_Maintenance\_Item c wp = new Equipment\_Maintenance\_Item c(Equipment c = equipmentId,

Maintenance\_Request c = requestId); return wp;

}

@istest

private static void testMaintenanceRequestPositive(){ Vehicle c vehicle = createVehicle();

insert vehicle;

id vehicleId = vehicle.Id;

Product2 equipment = createEq(); insert equipment;

id equipmentId = equipment.Id;

case somethingToUpdate = createMaintenanceRequest(vehicleId,equipmentId);

insert somethingToUpdate;

Equipment\_Maintenance\_Item c workP = createWorkPart(equipmentId,somethingToUpdate.id);

insert workP;

test.startTest(); somethingToUpdate.status = CLOSED; update somethingToUpdate; test.stopTest();

Case newReq = [Select id, subject, type, Equipment c,

Date\_Reported c, Vehicle c, Date\_Due c

from case

where status =:STATUS\_NEW];

Equipment\_Maintenance\_Item c workPart = [select id

from

Equipment\_Maintenance\_Item c Maintenance\_Request c =:newReq.Id];

where

system.assert(workPart != null); system.assert(newReq.Subject != null); system.assertEquals(newReq.Type, REQUEST\_TYPE); SYSTEM.assertEquals(newReq.Equipment c, equipmentId); SYSTEM.assertEquals(newReq.Vehicle c, vehicleId); SYSTEM.assertEquals(newReq.Date\_Reported c,

system.today());

}

@istest

private static void testMaintenanceRequestNegative(){ Vehicle C vehicle = createVehicle();

insert vehicle;

id vehicleId = vehicle.Id;

product2 equipment = createEq(); insert equipment;

id equipmentId = equipment.Id;

case emptyReq = createMaintenanceRequest(vehicleId,equipmentId);

insert emptyReq;

Equipment\_Maintenance\_Item c workP = createWorkPart(equipmentId, emptyReq.Id);

insert workP;

test.startTest(); emptyReq.Status = WORKING; update emptyReq; test.stopTest();

list<case> allRequest = [select id

from case];

Equipment\_Maintenance\_Item c workPart = [select id

from

Equipment\_Maintenance\_Item c Maintenance\_Request c = :emptyReq.Id];

where

system.assert(workPart != null); system.assert(allRequest.size() == 1);

}

@istest

private static void testMaintenanceRequestBulk(){ list<Vehicle C> vehicleList = new list<Vehicle C>(); list<Product2> equipmentList = new list<Product2>(); list<Equipment\_Maintenance\_Item c> workPartList = new

list<Equipment\_Maintenance\_Item c>();

list<case> requestList = new list<case>(); list<id> oldRequestIds = new list<id>();

for(integer i = 0; i < 300; i++){ vehicleList.add(createVehicle());

equipmentList.add(createEq());

}

insert vehicleList; insert equipmentList;

for(integer i = 0; i < 300; i++){ requestList.add(createMaintenanceRequest(vehicleLi

st.get(i).id, equipmentList.get(i).id));

}

insert requestList;

for(integer i = 0; i < 300; i++){

workPartList.add(createWorkPart(equipmentList.get( i).id, requestList.get(i).id));

}

insert workPartList;

test.startTest();

for(case req : requestList){ req.Status = CLOSED; oldRequestIds.add(req.Id);

}

update requestList; test.stopTest();

list<case> allRequests = [select id

from case

where status =: STATUS\_NEW];

list<Equipment\_Maintenance\_Item c> workParts = [select id

Equipment\_Maintenance\_Item c Maintenance\_Request c in: oldRequestIds];

from where

system.assert(allRequests.size() == 300);

}

}

MaintenanceRequestHelper.apxc

//code

public with sharing class MaintenanceRequestHelper { public static void updateworkOrders(List<Case>

updWorkOrders, Map<Id,Case> nonUpdCaseMap) { Set<Id> validIds = new Set<Id>(); For (Case c : updWorkOrders){

if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status == 'Closed'){

if (c.Type == 'Repair' || c.Type == 'Routine

Maintenance'){

}

}

}

validIds.add(c.Id);

if (!validIds.isEmpty()){

List<Case> newCases = new List<Case>(); Map<Id,Case> closedCasesM = new

Map<Id,Case>([SELECT Id, Vehicle c, Equipment c, Equipment r.Maintenance\_Cycle c,(SELECT

Id,Equipment c,Quantity c FROM Equipment\_Maintenance\_Items r)

Case WHERE Id IN :validIds]);

Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();

AggregateResult[] results = [SELECT Maintenance\_Request c,

MIN(Equipment r.Maintenance\_Cycle c)cycle FROM

FROM

Equipment\_Maintenance\_Item c WHERE Maintenance\_Request c IN

:ValidIds GROUP BY Maintenance\_Request c];

for (AggregateResult ar : results){ maintenanceCycles.put((Id)

ar.get('Maintenance\_Request c'), (Decimal) ar.get('cycle'));

}

for(Case cc : closedCasesM.values()){ Case nc = new Case (

ParentId = cc.Id, Status = 'New',

Subject = 'Routine Maintenance', Type = 'Routine Maintenance', Vehicle c = cc.Vehicle c, Equipment c =cc.Equipment c, Origin = 'Web',

Date\_Reported c = Date.Today()

);

If (maintenanceCycles.containskey(cc.Id)){ nc.Date\_Due c =

Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));

}

newCases.add(nc);

}

insert newCases;

List<Equipment\_Maintenance\_Item c> clonedWPs = new List<Equipment\_Maintenance\_Item c>();

for (Case nc : newCases){

for (Equipment\_Maintenance\_Item c wp : closedCasesM.get(nc.ParentId).Equipment\_Maintenance\_Items r){

Equipment\_Maintenance\_Item c wpClone =

wp.clone();

}

wpClone.Maintenance\_Request c = nc.Id; ClonedWPs.add(wpClone);

}

insert ClonedWPs;

}

}

}

MaintenanceRequest.apxt

//code

trigger MaintenanceRequest on Case (before update, after update) {

if(Trigger.isUpdate && Trigger.isAfter){ MaintenanceRequestHelper.updateWorkOrders(Trigger.New,

Trigger.OldMap);

}

}

## Test callout logic

WarehouseCalloutService.apxc

//code

public with sharing class WarehouseCalloutService {

private static final String WAREHOUSE\_URL = 'https://th- superbadge-apex.herokuapp.com/equipment';

public static void runWarehouseEquipmentSync(){ Http http = new Http();

HttpRequest request = new HttpRequest(); request.setEndpoint(WAREHOUSE\_URL); request.setMethod('GET');

HttpResponse response = http.send(request); List<Product2> warehouseEq = new List<Product2>(); if (response.getStatusCode() == 200){

List<Object> jsonResponse = (List<Object>)JSON.deserializeUntyped(response.getBody());

System.debug(response.getBody()); for (Object eq : jsonResponse){

Map<String,Object> mapJson = (Map<String,Object>)eq;

Product2 myEq = new Product2(); myEq.Replacement\_Part c = (Boolean)

mapJson.get('replacement');

myEq.Name = (String) mapJson.get('name'); myEq.Maintenance\_Cycle c = (Integer)

mapJson.get('maintenanceperiod');

myEq.Lifespan\_Months c = (Integer) mapJson.get('lifespan');

myEq.Cost c = (Decimal) mapJson.get('lifespan');

myEq.Warehouse\_SKU c = (String) mapJson.get('sku');

myEq.Current\_Inventory c = (Double) mapJson.get('quantity');

warehouseEq.add(myEq);

}

if (warehouseEq.size() > 0){ upsert warehouseEq;

System.debug('Your equipment was synced with the warehouse one');

System.debug(warehouseEq);

}

}

}

}

WarehouseCalloutServiceTest.apxc

//code @isTest

private class WarehouseCalloutServiceTest { @isTest

static void testWareHouseCallout(){ Test.startTest();

// implement mock callout test here Test.setMock(HTTPCalloutMock.class, new

WarehouseCalloutServiceMock()); WarehouseCalloutService.runWarehouseEquipmentSync(); Test.stopTest();

System.assertEquals(1, [SELECT count() FROM Product2]);

}

}

WarehouseCalloutServiceMock.apxc

//code @isTest

global class WarehouseCalloutServiceMock implements HttpCalloutMock {

global static HttpResponse respond(HttpRequest request){ System.assertEquals('https://th-superbadge-

apex.herokuapp.com/equipment', request.getEndpoint()); System.assertEquals('GET', request.getMethod()); HttpResponse response = new HttpResponse(); response.setHeader('Content-Type',

'application/json');

response.setBody('[{"\_id":"55d66226726b611100aaf741"," replacement":false,"quantity":5,"name":"Generator 1000 kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":" 100003"}]');

response.setStatusCode(200); return response;

}

}

## Test scheduling logic

WarehouseSyncSchedule.apxc

//code

global class WarehouseSyncSchedule implements Schedulable { global void execute(SchedulableContext ctx) {

WarehouseCalloutService.runWarehouseEquipmentSync();

}

}

WarehouseSyncScheduleTest.apxc

//code @isTest

public class WarehouseSyncScheduleTest {

@isTest static void WarehousescheduleTest(){ String scheduleTime = '00 00 01 \* \* ?'; Test.startTest(); Test.setMock(HttpCalloutMock.class, new

WarehouseCalloutServiceMock());

String jobID=System.schedule('Warehouse Time To Schedule to Test', scheduleTime, new WarehouseSyncSchedule());

Test.stopTest();

CronTrigger a=[SELECT Id FROM CronTrigger where NextFireTime > today];

System.assertEquals(jobID, a.Id,'Schedule ');

}

}