COMPARISON OF ASYNCHRONOUS APEX EXECUTION METHODS IN SALESFORCE

FUTURE :

FUTURE --> FUTURE (NO)

```
public class FutureExample1 {
   @future
   public static void futureMethode() {
      System.debug('I am Future method 1');

FutureExample2.futureMethode();

   }
}
```

ERROR:

	11/13/2025, 10:05 AM	Future	Failed	First error: Future method cannot be called from a future or batch method: FutureExample2.futureMethode()	

FUTURE --> 1 QUEUEABLE (YES)

```
public class FutureExample {
   @future
   public static void futureMethode() {
      System.debug('i am Future methode');

      System.enqueueJob(new FirstQueueable());
   }
}
```

FUTURE --> 2 QUEUEABLE (NO)

RUNTIME ERROR

```
public class FutureExample {
   @future
   public static void futureMethode() {

      System.debug('i am Future methode');

      System.enqueueJob(new FirstQueueable());
       System.enqueueJob(new SecondQueueable());
   }
}
```

ERROR :

First error: Too many queueable jobs added to the queue: 2

11/12/2025, 5:10 PM	Future	Failed	First error: Too many queueable jobs added to the queue: 2

FUTURE --> BATCH (NO)

RUNTIME ERROR

```
public class FutureExample {
   @future
   public static void futureMethode() {
      System.debug('i am Future methode');
     Database.executeBatch(new batchExample(), 5);
   }
}
```

ERROR:

First error: Database.executeBatch cannot be called from a batch start, batch execute, or future method.

11/12/2025, 5:14 PM	Future	Failed	First error: Database.executeBatch cannot be called from a batch start, batch execute, or future method.

FUTURE --> SCHEDULABLE (YES)

```
public class FutureExample {
   @future
   public static void futureMethode() {
      System.debug('i am Future methode');
      Datetime now = Datetime.now();
      Datetime runTime = now.addMinutes(2);
      String jobName = 'Run After 2 Minutes';
      System.schedule(jobName, runTime.format('ss mm HH dd MM ? yyyy'), new ScheduleExample());

   }
}
```

FUTURE -- > 2 SCHEDULE AND SAME TIME (NO)

```
public class FutureExample1 {
   @future
   public static void futureMethode() {
      System.debug('I am Future method 1');


       Datetime runTime1 = Datetime.now().addSeconds(2);
      String cronExp1 = runTime1.second() + ' ' + runTime1.minute() + ' ' + runTime1.hour() + ' ' +
                  runTime1.day() + ' ' + runTime1.month() + ' ? ' + runTime1.year();
      System.schedule('Job_After_2_Seconds', cronExp1, new ScheduleExample1());

      Datetime runTime2 = Datetime.now().addSeconds(2);
      String cronExp2 = runTime2.second() + ' ' + runTime2.minute() + ' ' + runTime2.hour() + ' ' +
                  runTime2.day() + ' ' + runTime2.month() + ' ? ' + runTime2.year();
      System.schedule('Job_After_7_Seconds', cronExp2, new ScheduleExample2());


   }
}
```

ERROR:

	11/13/2025, 11:06 AM	Future	Failed	First error: The Apex job named "Job_After_2_Seconds" is already scheduled for execution.

FUTURE --> 2 SCHEDULE DIFFERENT TIME (YES)

```
public class FutureExample1 {
   @future
   public static void futureMethode() {
      System.debug('I am Future method 1');


       Datetime runTime1 = Datetime.now().addSeconds(2);
      String cronExp1 = runTime1.second() + ' ' + runTime1.minute() + ' ' + runTime1.hour() + ' ' +
                  runTime1.day() + ' ' + runTime1.month() + ' ? ' + runTime1.year();
      System.schedule('Job_After_2_Seconds', cronExp1, new ScheduleExample1());


      Datetime runTime2 = Datetime.now().addSeconds(7);
      String cronExp2 = runTime2.second() + ' ' + runTime2.minute() + ' ' + runTime2.hour() + ' ' +
                  runTime2.day() + ' ' + runTime2.month() + ' ? ' + runTime2.year();
      System.schedule('Job_After_7_Seconds', cronExp2, new ScheduleExample2());
   }
}
```

QUEUEABLE

QUEUEABLE--> 1 FUTURE(YES)

```
public class FirstQueueable implements Queueable {
    public void execute(QueueableContext context) {
        System.debug('First Queueable Job Executed');
        FutureExample.futureMethode();

    }
}
```

QUEUEABLE--> MORE THAN FUTURE(YES)

```
public class FirstQueueable implements Queueable {
    public void execute(QueueableContext context) {
        System.debug('First Queueable Job Executed');
       FutureExample2.futureMethode2();
        FutureExample1.futureMethode();
    }
}
```

QUEUEABLE --> 1 QUEUEABLE (YES)

```
public class FirstQueueable implements Queueable {
    public void execute(QueueableContext context) {
        System.debug('First Queueable Job Executed');
       System.enqueueJob(new SecondQueueable());

    }
}
```

QUEUEABLE --> MORE THAN 1 QUEUEABLE (NO)

```
public class FirstQueueable implements Queueable {
    public void execute(QueueableContext context) {
        System.debug('First Queueable Job Executed');
       System.enqueueJob(new SecondQueueable());
       system.enqueueJob(new thirdQueueable());
    }
}
```

ERROR:

Action	Submitted Date ↓	Job Type	Status	Status Detail
	11/13/2025, 9:58 AM	Queueable	Failed	Too many queueable jobs added to the queue: 2

Too many queueable jobs added to the queue: 2

QUEUEABLE --> BATCH (YES)

```
public class thirdQueueable implements Queueable {
    public void execute(QueueableContext context) {
        System.debug('Third Queueable Job Executed');
        Database.executeBatch(new batchExample(), 20);
    }
}
```

QUEUEABLE --> MORE THAN 1 BATCH (YES)

```
public class thirdQueueable implements Queueable {
    public void execute(QueueableContext context) {
        System.debug('Third Queueable Job Executed');
        Database.executeBatch(new batchExample(), 20);
        Database.executeBatch(new batchExample2(),  20);
    }
}
```

QUEUEABLE -->SCHEDULE(YES)

```
public class FirstQueueable implements Queueable {
    public void execute(QueueableContext context) {
        System.debug('First Queueable Job Executed');
        Datetime runTime1 = Datetime.now().addSeconds(2);
        String cronExp1 = runTime1.second() + ' ' + runTime1.minute() + ' ' + runTime1.hour() + ' ' +
                    runTime1.day() + ' ' + runTime1.month() + ' ? ' + runTime1.year();
        System.schedule('Job_After_2_Seconds', cronExp1, new ScheduleExample1());

    }
}
```

QUEUEABLE --> MORE THAN 1 SCHEDULE (YES)

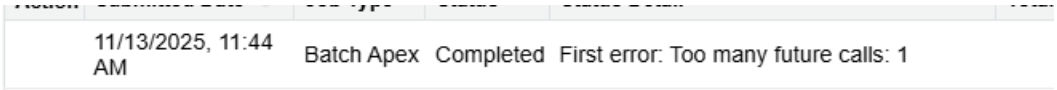
```
public class FirstQueueable implements Queueable {
    public void execute(QueueableContext context) {
        System.debug('First Queueable Job Executed');



        Datetime runTime1 = Datetime.now().addSeconds(2);
        String cronExp1 = runTime1.second() + ' ' + runTime1.minute() + ' ' + runTime1.hour() + ' ' +
                    runTime1.day() + ' ' + runTime1.month() + ' ? ' + runTime1.year();
        System.schedule('Job_After_2_Seconds', cronExp1, new ScheduleExample1());


         // After 7 seconds
        Datetime runTime2 = Datetime.now().addSeconds(7);
        String cronExp2 = runTime2.second() + ' ' + runTime2.minute() + ' ' + runTime2.hour() + ' ' +
                    runTime2.day() + ' ' + runTime2.month() + ' ? ' + runTime2.year();
        System.schedule('Job_After_7_Seconds', cronExp2, new ScheduleExample2());



    }
}
```

BATCH:

BATCH --> 1 FUTURE (NO)

```
public class batchExample implements Database.Batchable<sObject> {
    public Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator('SELECT Id, Name FROM Account');
    }


    public void execute(Database.BatchableContext bc, List<Account> scope) {
        system.debug(scope);
        system.debug('this is batch apex1 ');

    }

    public void finish(Database.BatchableContext bc) {

FutureExample1.futureMethode();
    }

}
```

ERROR:



	11/13/2025, 11:44 AM	Batch Apex	Completed	First error: Too many future calls: 1	

First error: Too many future calls: 1

BATCH --> MORE THAN 1 FUTURE (NO)

BATCH --. 1 QUEUEABLE(YES)

```
public class batchExample implements Database.Batchable<sObject> {
    public Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator('SELECT Id, Name FROM Account');
    }


    public void execute(Database.BatchableContext bc, List<Account> scope) {
        system.debug(scope);
        system.debug('this is batch apex1 ');

    }

    public void finish(Database.BatchableContext bc) {

System.enqueueJob(new FirstQueueable());
    }

}
```

BATCH --> MORE THAN 1 QUEUEABLE (NO)

```
public class batchExample implements Database.Batchable<sObject> {
    public Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator('SELECT Id, Name FROM Account');
    }
    public void execute(Database.BatchableContext bc, List<Account> scope) {
        system.debug(scope);
        system.debug('this is batch apex1 ');

    }

    public void finish(Database.BatchableContext bc) {

System.enqueueJob(new FirstQueueable());
        System.enqueueJob(new SecondQueueable());
    }

}
```

ERROR:

Action	Submitted Date ↓	Job Type	Status	Status Detail
	11/13/2025, 11:51 AM	Batch Apex	Completed	First error: Too many queueable jobs added to the queue: 2

First error: Too many queueable jobs added to the queue: 2

BATCH --> 1 BATCH (YES)

```
public class batchExample implements Database.Batchable<sObject> {
    public Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator('SELECT Id, Name FROM Account');
    }


    public void execute(Database.BatchableContext bc, List<Account> scope) {
        system.debug(scope);
        system.debug('this is batch apex1 ');

    }

    public void finish(Database.BatchableContext bc) {

Database.executeBatch(new batchExample2(),  10);
    }

}
```

BATCH --> MORE THAN 1 BATCH (YES)

```
public class batchExample implements Database.Batchable<sObject> {
    public Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator('SELECT Id, Name FROM Account');
    }


    public void execute(Database.BatchableContext bc, List<Account> scope) {
        system.debug(scope);
        system.debug('this is batch apex1 ');

    }

    public void finish(Database.BatchableContext bc) {

        Database.executeBatch(new batchExample2(),  10);
        Database.executeBatch(new batchExample3(),  10);
    }

}
```

BATCH --> SCHEDULABLE (YES)

```
public class batchExample implements Database.Batchable<sObject> {
    public Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator('SELECT Id, Name FROM Account');
    }


    public void execute(Database.BatchableContext bc, List<Account> scope) {
        system.debug(scope);
        system.debug('this is batch apex1 ');

    }

    public void finish(Database.BatchableContext bc) {

         Datetime runTime1 = Datetime.now().addSeconds(2);
        String cronExp1 = runTime1.second() + ' ' + runTime1.minute() + ' ' + runTime1.hour() + ' ' +
                        runTime1.day() + ' ' + runTime1.month() + ' ? ' + runTime1.year();
        System.schedule('Job_After_2_Seconds', cronExp1, new ScheduleExample1());

    }

}
```

BATCH --> MORE THAN 1 SCHEDULABLE (YES)

```
public class batchExample implements Database.Batchable<sObject> {
    public Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator('SELECT Id, Name FROM Account');
    }


    public void execute(Database.BatchableContext bc, List<Account> scope) {
        system.debug(scope);
        system.debug('this is batch apex1 ');

    }

    public void finish(Database.BatchableContext bc) {

         Datetime runTime1 = Datetime.now().addSeconds(2);
        String cronExp1 = runTime1.second() + ' ' + runTime1.minute() + ' ' + runTime1.hour() + ' ' +
                    runTime1.day() + ' ' + runTime1.month() + ' ? ' + runTime1.year();
        System.schedule('Job_After_2_Seconds', cronExp1, new ScheduleExample1());

        Datetime runTime2 = Datetime.now().addSeconds(7);
        String cronExp2 = runTime2.second() + ' ' + runTime2.minute() + ' ' + runTime2.hour() + ' ' +
                    runTime2.day() + ' ' + runTime2.month() + ' ? ' + runTime2.year();
        System.schedule('Job_After_7_Seconds', cronExp2, new ScheduleExample2());

    }

}
```

SCHEDULE

SCHEDULE --> 1 FUTURE (YES)

```
public class ScheduleExample1 implements Schedulable {
    public void execute(SchedulableContext sc) {
        system.debug('hi i am schedular class1');

FutureExample2.futureMethode2();
    }
}
```

SCHEDULE --> MORE THAN 1 FUTURE(YES)

```
public class ScheduleExample1 implements Schedulable {
    public void execute(SchedulableContext sc) {
        system.debug('hi i am schedular class1');

FutureExample2.futureMethode2();
        FutureExample3.futureMethode3();
    }
}
```

SCHEDULE --> QUEUEABLE(YES)

```
public class ScheduleExample1 implements Schedulable {
    public void execute(SchedulableContext sc) {
        system.debug('hi i am schedular class1');

System.enqueueJob(new FirstQueueable());
    }
}
```

SCHEDULE --> MORE THAN 1 QUEUEABLE (YES)

```
public class ScheduleExample1 implements Schedulable {
    public void execute(SchedulableContext sc) {
        system.debug('hi i am schedular class1');

System.enqueueJob(new FirstQueueable());
        System.enqueueJob(new SecondQueueable());
    }
}
```

SCHEDULE --> BATCH(YES)

```
public class ScheduleExample1 implements Schedulable {
    public void execute(SchedulableContext sc) {
        system.debug('hi i am schedular class1');
   Database.executeBatch(new batchExample(), 20);
    }
}
```

SCHEDULE --> MORETHAN 1 BATCH(YES)

```
public class ScheduleExample1 implements Schedulable {
    public void execute(SchedulableContext sc) {
        system.debug('hi i am schedular class1');
   Database.executeBatch(new batchExample(), 20);
        Database.executeBatch(new batchExample2(),  20);
    }
}
```

SCHEDULE --> 1 SCHEDULE(YES)

```
public class ScheduleExample1 implements Schedulable {
    public void execute(SchedulableContext sc) {
        system.debug('hi i am schedular class1');

       Datetime runTime2 = Datetime.now().addSeconds(2);
        String cronExp2 = runTime2.second() + ' ' + runTime2.minute() + ' ' + runTime2.hour() + ' ' +
                    runTime2.day() + ' ' + runTime2.month() + ' ? ' + runTime2.year();
       System.schedule('Job_After_7_Seconds', cronExp2, new ScheduleExample2());
    }
}
```

SCHEDULE --> MORE THAN SCHEDULE (YES)

```
public class ScheduleExample1 implements Schedulable {
    public void execute(SchedulableContext sc) {
        system.debug('hi i am schedular class1');

       Datetime runTime2 = Datetime.now().addSeconds(7);
        String cronExp2 = runTime2.second() + ' ' + runTime2.minute() + ' ' + runTime2.hour() + ' ' +
                    runTime2.day() + ' ' + runTime2.month() + ' ? ' + runTime2.year();
       System.schedule('Job_After_7_Seconds', cronExp2, new ScheduleExample2());


        Datetime runTime3 = Datetime.now().addSeconds(13);
        String cronExp3 = runTime3.second() + ' ' + runTime3.minute() + ' ' + runTime3.hour() + ' ' +
                    runTime3.day() + ' ' + runTime3.month() + ' ? ' + runTime3.year();
       System.schedule('Job_After_13_Seconds', cronExp3, new ScheduleExample3());
    }
}
```

	1 FUTURE	MORE THAN 1 FUTURE	1 QUEUEABLE	MORE THAN 1 QUEUEABLE	1 BATCH	MORE THAN 1 BATCH	1 SCHEDULE	MORE THAN 1 SCHEDULE
FUTURE	NO ERROR: First error: Future method cannot be called from a future or batch method SOLUTION: Prefer Queueable Apex	NO	YES	NO ERROR: First error: Too many queueable jobs added to the queue: 2 SOLUTION: One Queueable → One Next Queueable (chain sequentially).	NO ERROR: First error: Database.executeBatch cannot be called from a batch start, batch execute, or future method. SOLUTION: Use Queueable Apex or Scheduled Apex for chaining.	NO	YES	YES
QUEUEABLE	YES	YES	YES	NO ERROR: Too many queueable jobs added to the queue: 2 SOLUTION: One Queueable → One Next Queueable (chain sequentially).	YES	YES	YES	YES
BATCH	NO ERROR: First error: Too many future calls: 1 SOLUTION: Use Queueable Apex	NO	YES	NO First error: Too many queueable jobs added to the queue: 2 SOLUTION: One Queueable → One Next Queueable (chain sequentially).	YES	YES	YES	YES
SCHEDULE	YES	YES	YES	YES	YES	YES	YES	YES