

# Introduction to Modern Software Development

ORACLE

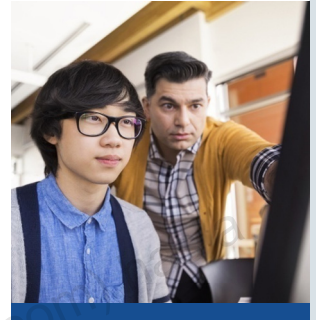
Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Kevin Gómez González (kev.gomez97@hotmail.com) has a non-transferable license to use this Student Guide.

## Objectives

After completing this lesson, you should be able to:

- Describe the concepts of the Scrum methodology
- Create user stories to describe requirements
- Estimate stories and team capacity
- Describe the fundamentals of test-driven development
- Create assertions in your code
- Design and create code based on tests
- Describe what Oracle Developer Cloud Service is
- List the main components of Oracle Developer Cloud Service
- Explain a typical workflow for using Oracle Developer Cloud Service

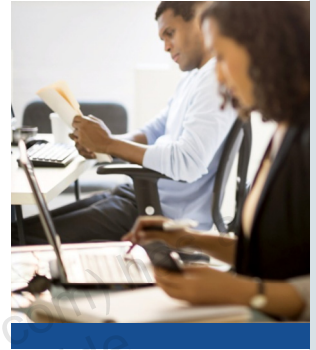


ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

## Topics

- Introduction to Agile Development and Scrum
- Introduction to test-driven development
- Introduction to Oracle Developer Cloud Service



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Kevin Gómez González (kev.gomez97@hotmail.com)  
non-transferable license to use this Student Guide.

## Software Development Process

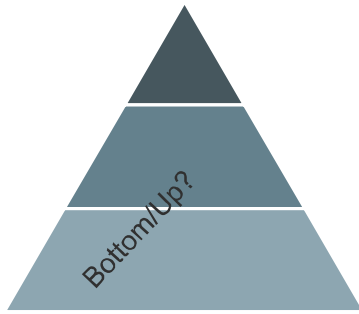
**ORACLE**

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

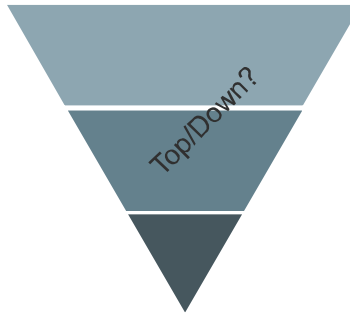
Building software is a complex process that involves many steps and considerations. You need a team of developers, analysts, testers, architects, and so on. You have to translate your client's ideas into requirements that can be developed by your team, and then estimate how long would it take your team to develop the project. Finally, you must make sure that the solution is sufficient and high quality by reducing or eliminating software defects.

From the simplest system to the most complex, it is a good idea to follow a structured process that allows the project to be properly tracked and measured. Multiple methodologies can be used to help ensure that software projects are successfully developed. In this course, Scrum methodology is used.

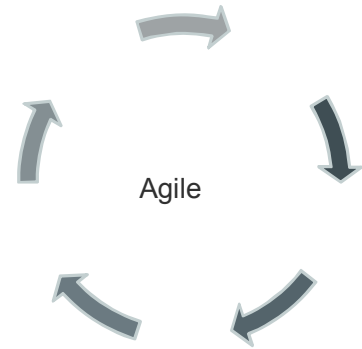
## Agile Methodologies



- Analyze requirements.
- Then build up the system.



- Start building the system.
- Find out if you met the criteria along the way.



- Start with requirements.
- Then build the system incrementally.
- Analyze just what you need to get started, build, then analyze again.

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Agile effectively combines bottom-up and top-down methodologies, but Agile seeks to change the way software is developed by putting more emphasis on the software development than on the process, or on comprehensive documentation, or even on planning.

The *Manifesto for Agile Software Development* states:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in bottom-up and top-down development, we value the Agile process more.

## Scenario: Online Shopping

"I want a website where I can sell my products online."

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

From a simple statement like the one above, an entire system can be built.

Obviously, you need to question your client about specific requirements, such as:

- Does the stock change frequently?
- Do we want to control inventory?

The problem with this process is that finding the right questions can take a great amount of time and, as you start asking questions, you will find more problems and more questions that need to be asked.

Agile methodologies promote the idea that you need to ask questions until you have a good idea about the size of the project without investing too much time. Usually, you want to set acceptance criteria for the project and then work out the details from there.

## User Story

Define a quick requirement using the following template:

As a [USER TYPE], I want to [REQUIREMENT], to [BUSINESS VALUE].

As a customer, I want to see a home page with the company logo, to know where I am.

As an owner, I want to see the purchases made by my customers to fulfill them

As a developer, I want to set up the development environment to prepare it for the team

As a customer I want to add an item to my shopping cart to buy it later

As a customer, I want to see items as a list to view the catalog

As a customer, I want to see items as a list to view the catalog

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

User stories are descriptive requirements for the system that should be small enough to fit in a simple sentence.

The sentence should state:

- The requirement
- Who needs the requirement
- What business value the requirement has

Later, stories can contain more information, but the basic sentence should remain unchanged as it drives the entire story development. From that one sentence, you can expand the acceptance criteria and development details.

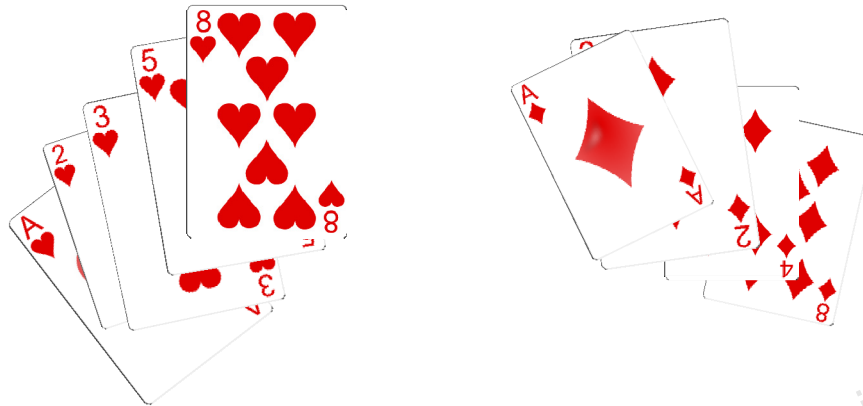
### Scenario

Some user stories:

- As a customer, I want to see a home page with the company logo, to know where I am.
- As a customer, I want to see items as a list, to view the catalog.
- As a customer, I want to see the details of an item, to get more information about it.
- As a customer, I want to add an item to my shopping cart, to buy it later.
- As a developer, I want to set up the development environment, to prepare it for the team.
- As an owner, I want to see the purchases made by my customers, to fulfill them.

## Estimating Stories: Planning Poker

- Estimate how long each user story will take to implement based on its difficulty.
- Assign the easiest tasks a value of 1, and then assess and rate the other stories against the easiest one. How much more difficult are the others?



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Stories should have a value that estimates their duration or difficulty.

Historically, use cases and stories were estimated in hours but, as systems became more complex, hour estimation proved insufficient and inaccurate, and point-based estimations were adopted.

Point-based estimations state that you must find the easiest story among your backlog. That story becomes your base 1pt story. From there, the team is asked if the next story is as difficult as the first story (1pt), twice as hard as difficult as the first story (2pt), three times as hard as difficult as the first story (3pt), and so on.

This is usually done with poker cards, where each member holds an ace (1pt), and cards 2, 4, and 8. The team is asked to assess the difficulty of each story, after which each member chooses a card in secret and then reveals the cards at the same time. If there is a consensus, the voted points are assigned. If there is a dispute, the point value is negotiated, using their votes.

Alternatively, you can use Fibonacci style progressions for points: 1,2,3,5,8.

### Scenario

- As a customer, I want to see a homepage... (1 pt)
- As a customer, I want to see items as a list ... (2 pts)
- As a customer, I want to see the details of an Item ... (4 pts)
- As a developer, I need to set up a basic web server ... (4 pts)



## Sprint

- Add user stories to a sprint.
- A sprint is a “time box,” a fixed period of time allocated to complete a set of stories.
- A sprint contains multiple user stories, each of which represents a requirement that provides value to the system.
- Sprints should be as short as possible, usually lasting 1 week to 1 month.
- A sprint should deliver working software by the end.

**ORACLE**

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

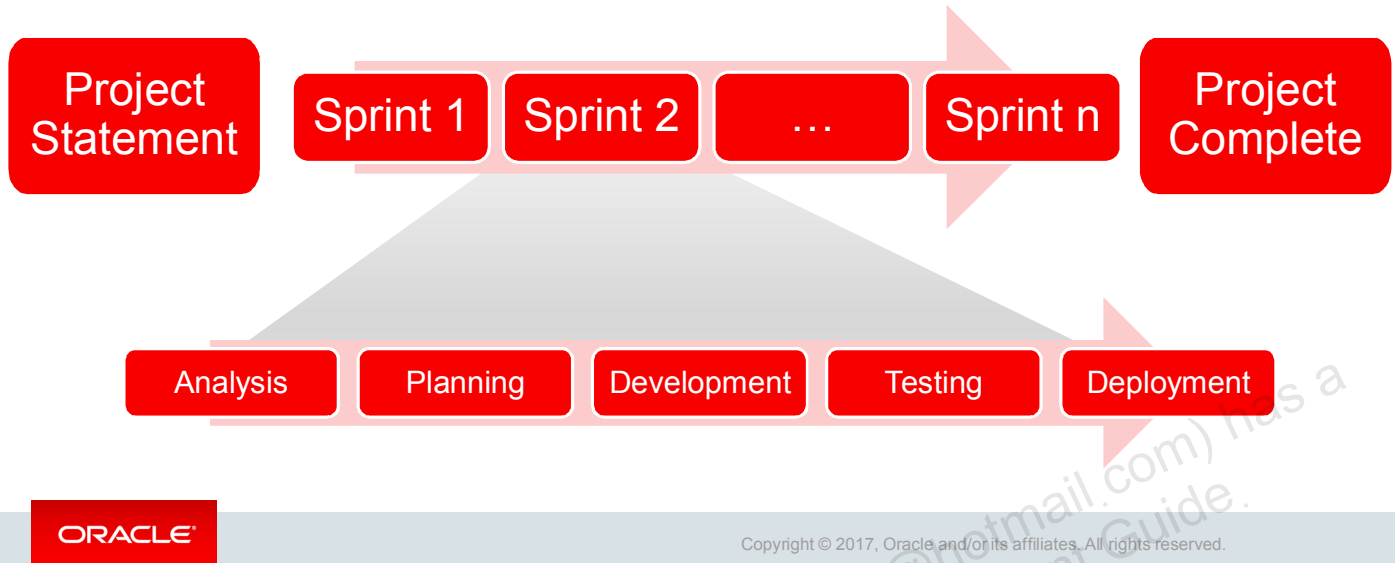
A sprint is the span of time during which you develop stories.

### Scenario

Sprint length: One or two weeks.  
Need to accelerate results to find possible problems faster

## Scrum

Scrum is an Agile development methodology that allows incremental product development by dividing it into small chunks of functionality.



Scrum is a subset of Agile. It is a lightweight process framework for Agile development, and the most widely-used one. The main goal of Scrum is to make the software development process more Agile. Instead of trying to foresee the entire system and working towards a big goal, you work on smaller and more manageable chunks, or sprints.

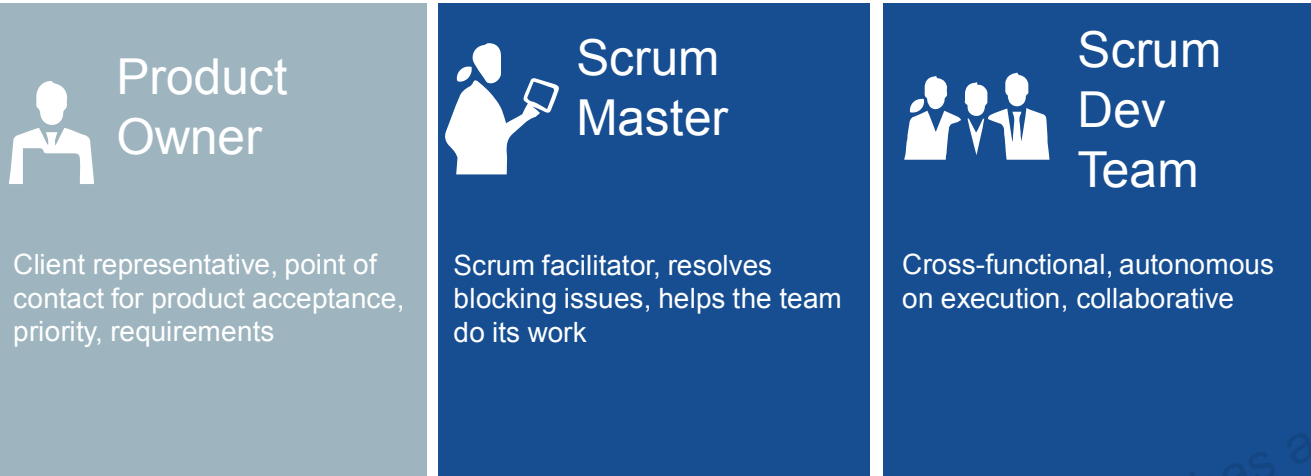
A system may or may not have a definitive scope. You can either define a fixed point when you are satisfied and complete the system or you can define a more flexible goal. In either case, the goal and functionality your system might have are readjusted frequently with each sprint.

Constant feedback from the client or product owner is required. Adaptation to change is one of the key values of Scrum.

### Scenario

We have a certain idea on when the project is done: Users are able to buy online from us. Details are filled later.

## Roles



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Scrum methodology addresses three user roles:

- The **product owner** represents your client. He makes decisions about requirements, prioritizes stories, sets acceptance criteria, and, most importantly, works with the team to answer questions that arise during development.
- The **Scrum master** oversees the process to make sure that stories are developed and that the product owner is pleased with the result. He helps the team by removing issues that block their progress. He is not a manager or a boss, but rather a facilitator, whose primary objective is to help the team work without impediments.
- The **Scrum development team** is a cross-functional set of expert individuals that develop stories into the final product. The team is not divided in different responsibilities; instead, the entire team is responsible for every aspect of story development. The team consists of individuals that develop, test, analyze, and estimate stories. Their responsibility and investment in the project is complete, because they help and participate in every aspect of the project.

## Scrum Meeting

Communication is key!

Have a quick meeting (15 minutes, at most) every day and ask the following questions to each team member:

- What did you do yesterday?
- What are you going to do today?
- Do you have any blockers or have you found any issues?

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

In Scrum, communication is key. To facilitate team communication and collaboration, a daily meeting should be held. Each and every member of the team should be asked the three questions shown in the slide. The Scrum Master can then follow up any issues to ensure that the team can develop their stories efficiently.

## Sprint Backlog

A scrum board is used to track the progress of user stories. Each story is placed in a column which reflects its status.

### Backlog

As a customer, I want to see a home page with the company logo to know where I am

### In Progress

As a customer, I want to see items as a list to view the catalog

### Done

As a developer, I want to set up the development environment to prepare it for the team

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

All committed stories for the sprint are recorded on the scrum board. Stories are picked by the team and advanced until they are done. The goal of the team is to get all stories done by the end of the sprint.

Your scrum board can look different and have more steps than the one presented in the slide. You can add Testing, Design, or any other step that your process requires. The slide presents the simplest steps that you should have in a project.

“Backlog” contains items that haven't been started.

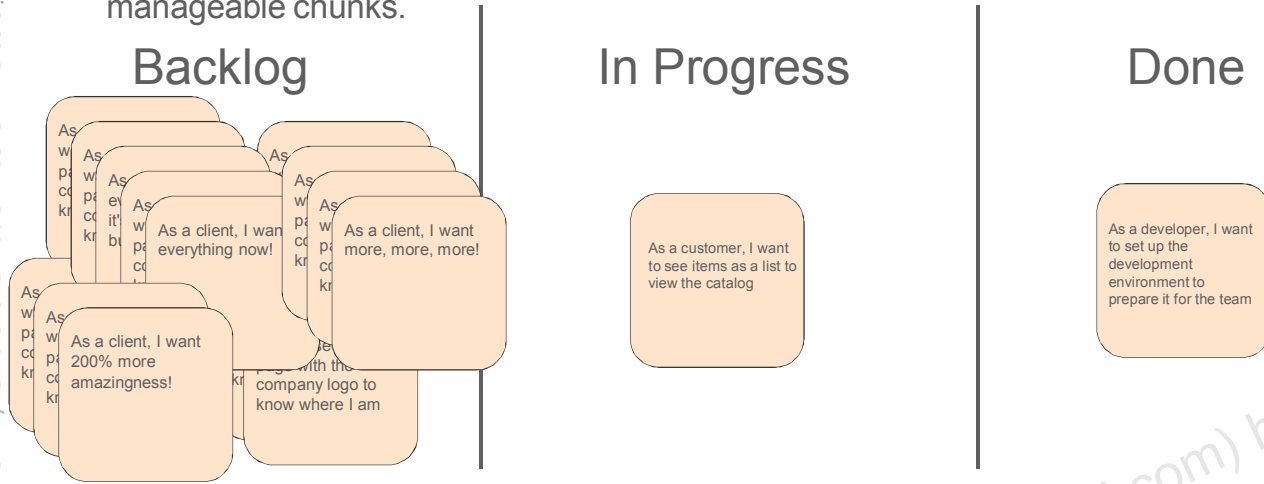
“In-Progress” means that work is being done on an item.

“Done” means an item has all its criteria met and fully complies with the acceptance criteria.

You can track stories or tasks by splitting stories into smaller tasks and moving them into the Backlog column.

## Planning Unrealistically

- It is very unlikely you can accomplish everything in one sprint.
- Instead of trying to foresee the entire system, sprints seek to create smaller, more manageable chunks.

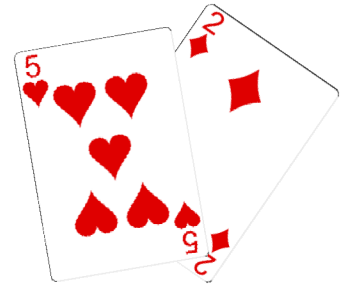
**ORACLE**

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

## Planning a Sprint Properly

Think about how much could be done in the allocated time. Add some stories and the number in points. The sum of the points is your team capacity, which could change, as you see fit.

- All stories are done and no extra time is left:  
The capacity is accurate.
- Some stories are left undone:
  - Story estimation is wrong.
  - Sprint capacity was exceeded; lower it.
- There is free time at the end:
  - Story estimation is wrong.
  - Sprint capacity was underused; increase it.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

At the beginning of a sprint, you plan which stories you will develop. At the end of a sprint, you evaluate the done story points. If you had enough time to build all the stories, the story point value is your team capacity. In the next sprint, you should plan the same number of points.

If you had extra time at the end of the sprint, you may have incorrectly estimated stories. You should check whether the effort required for the stories reflect their story points. If you find some discrepancies, adjust the values. If you had extra time and stories were correctly estimated, then do more points in the next sprint.

If you couldn't finish some stories, you probably underestimated some stories. This usually happens if low value stories were left out, even though they were developed from the start of the sprint. Readjust the story values. In the next sprint, plan to do less points, because the team is over capacity.

Don't expect the process to be perfect, especially in the first few sprints. The process requires constant adjustment to find an accurate velocity. Team changes will alter this value. Developer expertise and knowledge of the system is a very important factor to be considered.

### Scenario

Sprint 1: 7pts

- 4pt Display item list
- 1pt Home Page
- 2pt Set up

## Definition of Done

- Write stories with criteria and enough information so the task can be estimated and understood correctly.
- Stories can be expanded during development and between sprints, but never after they are completed.
- A story is "done" when its acceptance criteria is met and it is part of the active deployment of the system.
- A story is **not** done if it...
  - Is not the final code
  - Has not been integrated into the system
  - Was not tested
  - Does not meet all the acceptance criteria

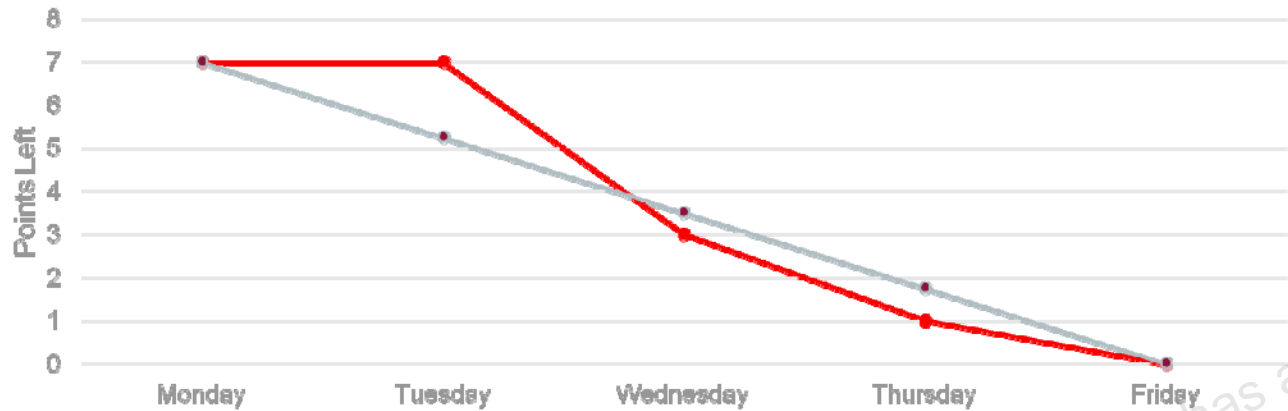
The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.



## Sprint Progress

Tasks are updated every day. Progress is measured based on how many points were completed. This chart is called a “burndown” chart.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Tracking progress in Scrum must be done daily, preferably while the Scrum meeting is in place. The team moves the stories as they are done and the Scrum Master updates the chart to reflect the progress of the sprint.

The estimated burndown is a line that goes from the high on the first day of the sprint and the total points to be done, to the low on the final day and 0 points.

The Scrum Master can see potential problems in the sprint if the burndown chart and the estimate diverge too far from each other. They should follow similar paths.

## Sprint Results

At the end of the sprint:

- Showcase the product with the product owner
- Adjust sprint capacity
- Identify new stories
- Estimate new stories
- Re-estimate existing stories, if needed
- Drop unneeded stories
- Select stories for the next sprint
- Start the next sprint



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

At the end of the sprint, you showcase the product, adjust the velocity based on “done” points, identify new stories, and analyze and estimate stories.

The product owner and the team discuss the next sprint stories, the priority, and the estimated value of stories.

## When Is the Project Done?

- When you end a sprint and no more stories are left to be developed
- When the product owner is satisfied
- When there is no more time
  - Scrum optimizes development based on the business value of stories.
  - The most beneficial stories should have already been implemented.
  - Lower priority stories might never be implemented.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

If the product owner has no more requirements at the end of a sprint, the system is done.

If the project has a strict time frame, you would have first developed the most valuable parts of the system, based on decisions made by the product owner, and you would have a clear view of what is being left out or what has been added. In any case, by building the highest priority stories and having a clear picture of progress of the system, you can provide the best value in the least amount of time.

## Other Tools and Techniques

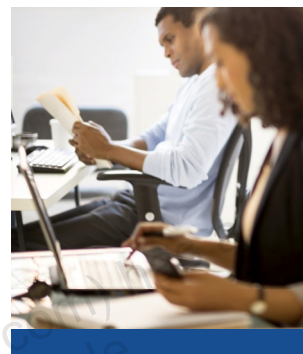
- Test-driven development (TDD)
  - Helps write efficient, safe code by writing tests first and implementations later
- Continuous Integration (CI)
  - Leverages the build process from the developers by running constant builds and running tests with every code change
- Continuous deployment
  - Helps system administrators with the deployment of a test version of the system in a controlled environment where it can be tested for acceptance by the product owner
- Pair programming
  - Helps developers find better solutions by pairing programmers while developing a story. Used with TDD, one person writes the test, the other fixes the code, and then roles are switched. This helps greatly with new team member mentoring.

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

## Topics

- Introduction to Agile Development and Scrum
- Introduction to test-driven development
- Introduction to Oracle Developer Cloud Service



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2014, Oracle and/or its affiliates.

Kevin Gómez González (kev.gomez97@hotmail.com)  
non-transferable license to use this Student Guide.

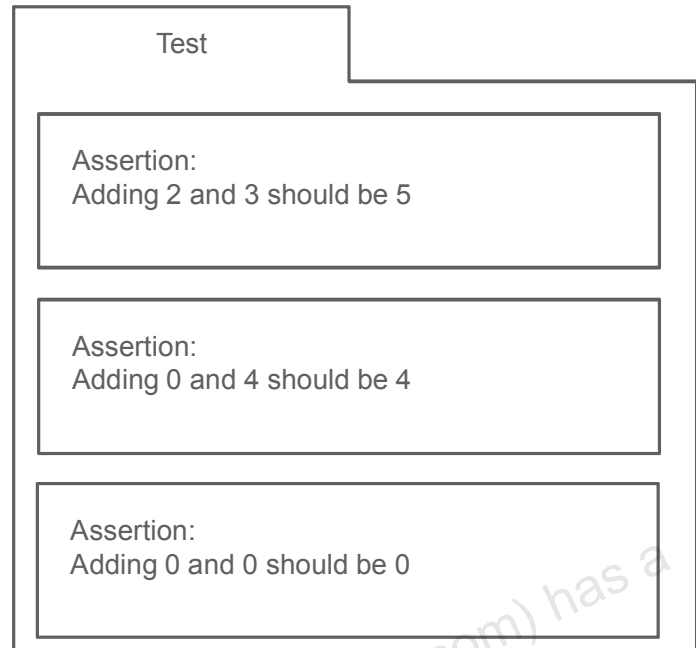
## What Is a Test?

A test is a set of assertions that check whether some functionality works the way it should.

An assertion is a simple comparison.

A test should be deterministic.

**The same inputs should yield the same results.**



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Tests are used to verify that a system works correctly. Historically, tests were written once the system was done, to validate its functionality and to provide some safety when making changes, because any change that breaks a test means that the change is not good.

Unfortunately writing the test after the system is done leads to uncovered scenarios and sometimes writing tests proves to be too difficult because writing a test on a full system can require a lot of effort. This problem led to a change in how tests are considered, and by testing before writing any actual code, a test now defines the functionality of a piece of software.

Writing tests first simplifies the process of writing tests, because you worry only about the particular functionality of a piece of the system instead of trying to test the system entirely.

Tests consists of: a setup, where you set the initial conditions of the system, a set of assertions, where you test functionality and return values, and finally a teardown where you clean up any used resources.

In its most basic form, a test is just a set of comparisons for known values over idempotent operations.

## Advantages

- Write the least amount of code for the functionality that you want.
- Tests provide a safety net when you modify your code.
  - If a modification breaks a test, there is likely a problem.
- Tests can be automated to check an entire system.
- Test-driven development aligns with Scrum.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

When you use test-driven development, you gain the advantage of having your code tested and keeping it working over time. Writing tests first reduces chances of over-engineering the code. You write just enough code to make the tests work.

Tests can be in different parts of the system and can be made to run automatically. A best practice is to have tests automated; this is possible by using testing frameworks in your system. Most systems allow you to separate the test code from the system code allowing you to run the tests and generate a final deliverable artifact without the tests being bundled.

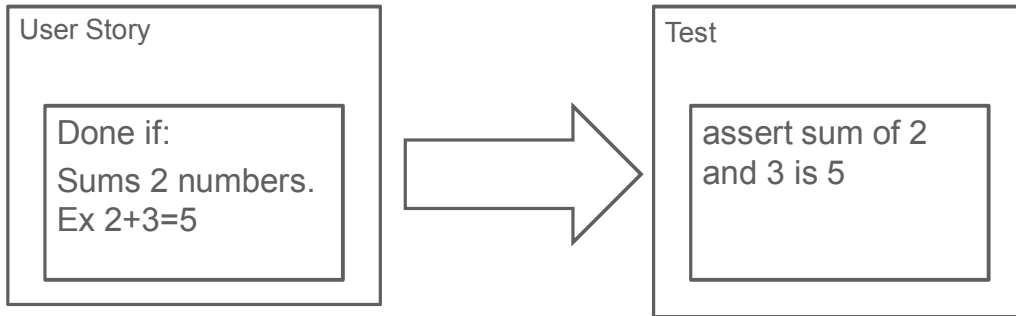
In Scrum, your stories should be short. Test-driven development helps you identify the size of a big story and relies on very small deliverables.

When you write a test, you define the expected functionality that should never change if same conditions are met. A function that receives the same parameters and has the same state should always return the same value.

Even functions that are not supposed to return the same value should meet this criteria, for example, a random function. Generating a random can be tested as well, you can run the function a thousand times and in those values set some criteria of how distinct they have to be. To test a function that requires a random number, make the random number a parameter instead, and then once again the function will have a deterministic behavior.

## Test-Driven Development and Scrum

- A user story is represented by a set of tests.
- The acceptance criteria is described as test assertions.
- Scrum short cycles work well with TDD short processes.

**ORACLE**

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

In Scrum, a user story usually translates to a test or set of tests.



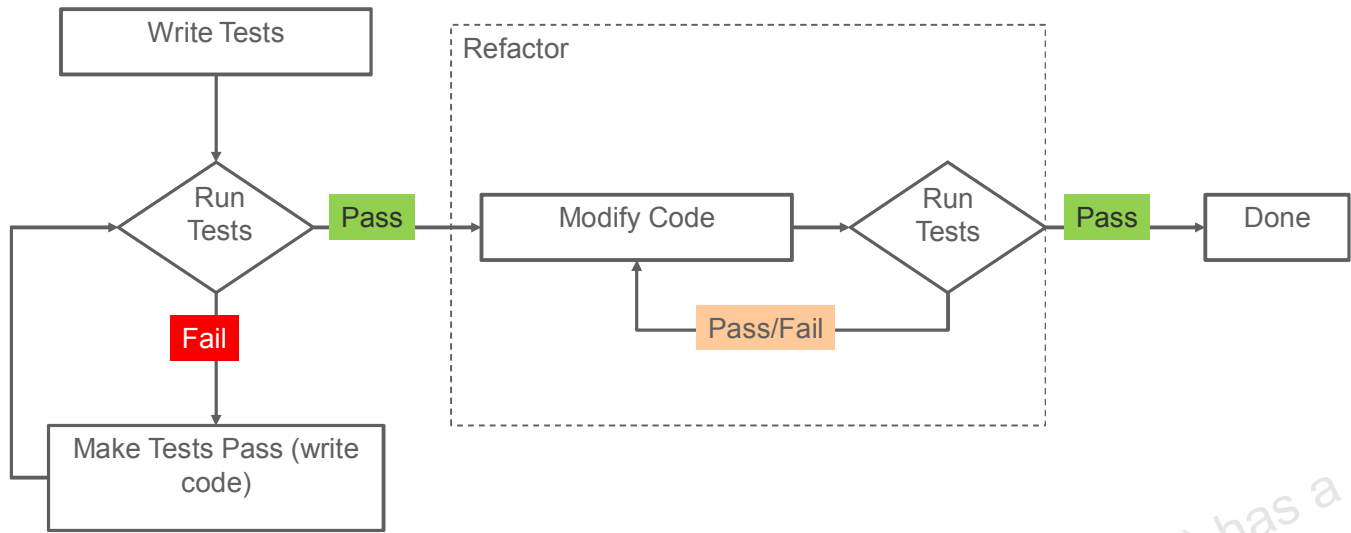
## How to Do Test-Driven Development

1. Think about what your code must do.
2. Write the test based on what you want the code to do.
3. Run the test (and it will fail).
4. Fix the test by writing the code.
5. Run the test until it works.
6. Refactor the code until you are satisfied.
7. Repeat step 5.

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

## Red, Green, Refactor



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

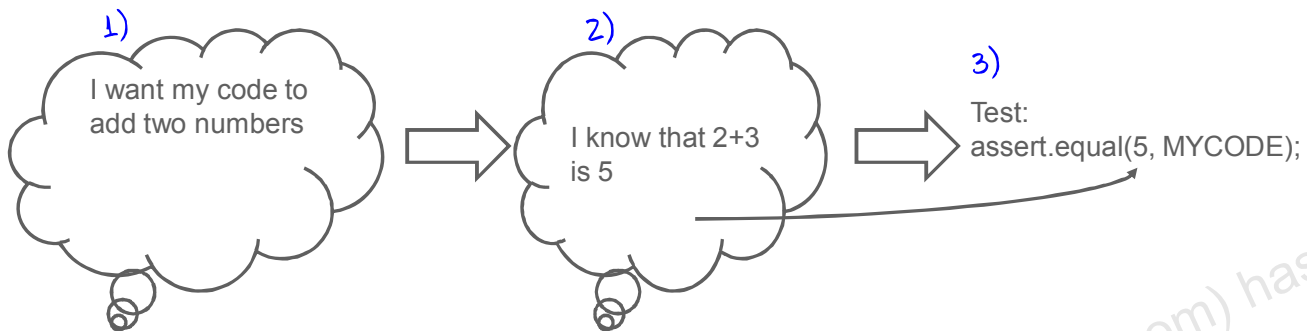
The process of writing software could be described as the RED, GREEN, REFACTOR cycle.

- You begin writing a test and running it. There will inevitably be failures, as there is no actual code yet. You are in RED, since the system doesn't comply with the tests.
- You will write code as necessary, without modifying the tests, until the test passes. You are in GREEN now, because tests do pass.
- You will REFACTOR the code, making it more readable and improving it. You modify code to remove duplicates, find optimizations, try new approaches, etc. You have to keep your code GREEN but you might move to RED at some point, because changes can negatively alter the system.

During this process, is important to keep a copy of the last GREEN state of the code, so you can undo your changes to the last working state.

## Develop Tests First

1. Think about what you want your code to do.
2. Find a specific scenario where your code will produce a known answer.
3. Write the test for that scenario (assertions).

**ORACLE**

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

So, how do you write the tests first? You do so by answering the question, "What do I want my code to do?" Then you write the assertions that you think you need. You can write as many scenarios as you like.

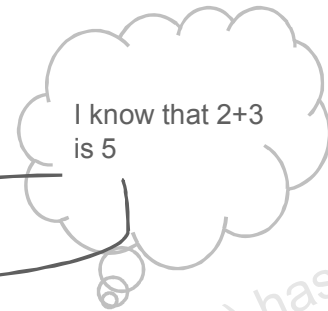
## Fix the Test

- After you know the assertions, begin fixing tests using those assertions.
- Write code and run the tests until all of your tests pass.
- If, at some point, you get stuck, you can delete everything and start again.
- Your goal is to make the test pass, not necessary to have optimal code.

```
assert.equal(5, ???);
```



```
assert.equal(5, 2+3);
```

**ORACLE**

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

## Modify the Code

- Optimize your solution:
  - Remove duplication
  - Condense lines of code
  - Add flexibility
  - Add more tests
  - Try new things

```
assert.equal(5, a+b);
```

```
var a=2;
var b=3;
assert.equal(5, a+b);
```

### DO NOT REMOVE TESTS

If a test has passed, it should keep passing after modifying the code.

```
assert.equal(5, sum(a,b));
```

```
assert.equal(5, sum(a,b));
function add(a,b){ return a+b;}
```

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Now that your test has passed, you can modify your code to reach an optimal solution and rewrite sections to improve its structure.

You should *not* remove tests. You can optimize them, but you should refrain from removing test cases because, if you do, you might be removing key scenarios.

If you maintain existing code that was used with TDD, you could face a scenario in which you need to modify an existing function. In this scenario, use the tests as a guideline of what functions should do and keep all tests passing while you make modifications.

## Descriptive Tests

- Tests and assertions should be descriptive.
- An assertion should be written with a description of what a function is supposed to do.
- In case of a failure, provide a message that describes what failed. This will help developers fix the problem.
- Descriptions are written in the form of “should” sentences.

```
assert.equal(5, sum(2,3), "Should add 2 and 3 to equal 5");  
assert.equal(20, mult(4,5), "Should mult 4 and 5 to equal 20");  
assert.throws(divide(10,0), "Should throw an error dividing by 0");
```

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

## Complicated Tests = A Problem with Requirements

If a test is too complicated, you could have two possible problems:

- A requirement is not small enough.
  - Redefine the requirement to make it smaller.
- The tested functionality does too much.
  - Divide your functionality into smaller chunks.

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

A problem can be hard to test; this can happen because user stories do too much or because you are over-engineering the problem. In these cases, consider splitting the problem into smaller, easier to handle pieces.

## Scenario: Home Page

### User story:

I want a program that greets the user by his or her name and displays his or her horoscope.

\_\_\_\_\_

### Test:

- Assert the user is greeted by name
- Assert the horoscope for the birthdate of the user is displayed

The problem in this slide seems to describe two different functions.

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.



## Scenario: Home Page – Two Stories

Story 1:

Greet user by name

Test:

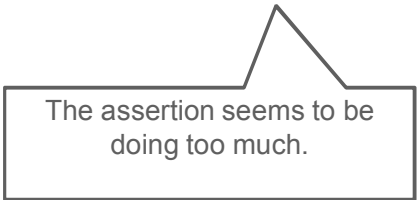
- Assert the greeting generated with username

Story 2:

Display user horoscope

Test:

- Assert horoscope by birthdate of the user



The assertion seems to be doing too much.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

First, split the story into smaller stories.

In the example, the stories describe two different requirements:

- Greeting the user based on his name
- Displaying a horoscope for the user, based on his birth date

## Scenario: Home Page – Two Stories and Smaller Tests

### Story 1:

Greet user by name

#### Test:

- Assert the greeting generated with username

### Story 2:

Display user horoscope

#### Test:

- Assert that the user date is captured
- Assert that the horoscope is correct by date

Remove the dependency on user specifics. Function gets horoscope by date.

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Then, split the over-engineered functionality.

In this scenario, it doesn't make sense to have one function request user data to transform to a date, and then get the horoscope using that date.

It should be done with two separate functions:

- The first function gets the user data and transforms it into a date.
- The second function receives a date and gets a horoscope.

After this is done, you can combine the two functions in your system to provide all of the requested functionality.

The advantage to this approach is that individual pieces can be tested and can be reused.

## Discussion

What would you do if your test case involves external factors or code,  
Such as a user interface or third-party services, where you do not control the code?

How do you test your code when it depends on third-party libraries or resources that do not  
provide a deterministic value, such as database access?

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

What if there are parts of your system that you cannot control while testing?  
How do you test when there is external code?  
How do you test when there is a dependency on external data?

## Advanced Test-Driven Development

- Assert relevant values, do not assert everything.
- Mock dependencies.
- Mock resources.
- Spy on external code and resources to verify that your code utilizes them correctly.
- Automate testing as part of your development process.
- Generate code coverage reports.
- Code coverage is the percentage of code that was executed in your tests.

## Mocking

A mock is a simulated object that replaces the actual one and is used in place of dependencies and external resources to make the code deterministic and allow inspection and control.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

A technique called mocking is often used when tests involve external factors. Mocking allows you to replace parts of your system with simulated parts that will return predictable values without causing any repercussions on the system.

Mocking is used with external services or pieces of software, but it can also be used in parts of your own system if the risk of using the actual parts proves to be too big. For example, if you have a function that writes random numbers to a file, you might want to mock that function because writing to the file system every time the test is invoked is troublesome because it might fill the disk.

Although you want to mock the function, you want to verify that it was invoked. You can spy on mocked objects and check what invocations were called on them. All of these features could be written as part of the test, but there are already testing frameworks that include mocking and spying capabilities. For now, just know that these techniques are available. The next slide includes pointers to resources that provide more information about mocking and testing.

## Additional Resources

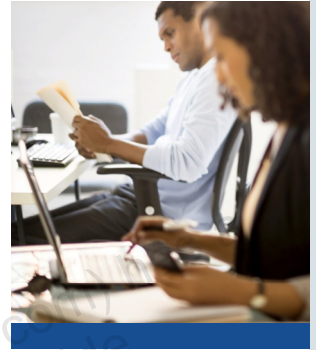
Topic	Website
Test Driven Development: By Example	Author: Kent Beck Editorial: Addison-Wesley Professional
Node assert package	<a href="https://nodejs.org/api/assert.html">https://nodejs.org/api/assert.html</a>
Mocha testing library	<a href="http://mochajs.org/">http://mochajs.org/</a>
Sinon JS Mocking and Stubbing library	<a href="http://sinonjs.org/">http://sinonjs.org/</a>

**ORACLE**

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

## Topics

- Introduction to Agile Development and Scrum
- Introduction to test-driven development
- Introduction to Oracle Developer Cloud Service



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2014, Oracle and/or its affiliates.

Kevin Gómez González (kev.gomez97@hotmail.com)  
non-transferable license to use this Student Guide.

## What Is Oracle Developer Cloud Service?

Oracle Developer Cloud Service is a set of tools in Oracle Cloud that enable you to manage software projects. These tools allow you to:

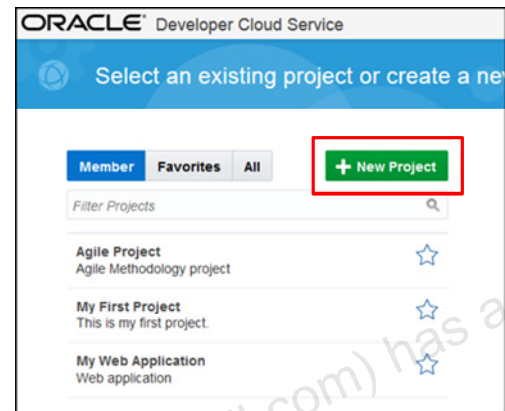
- Create a project to hold your application
- Manage your project's source code
- Perform Agile development using issue tracking and sprints
- Ask team members to review your code, then merge it after they review it
- Build, test, and deploy your application
- Document your project using a wiki
- Administer the project

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

## Projects and the Welcome Page

- You organize your code in a **project**.
- When you first start Oracle Developer Cloud Service, you land on the Welcome page, which contains a project list.
- Click the New Project button to create a project.

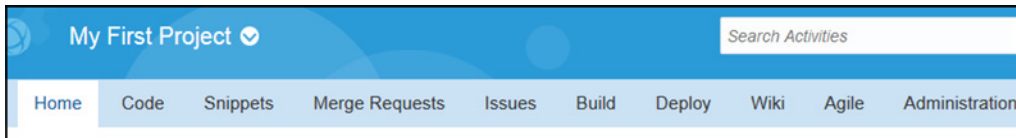


Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

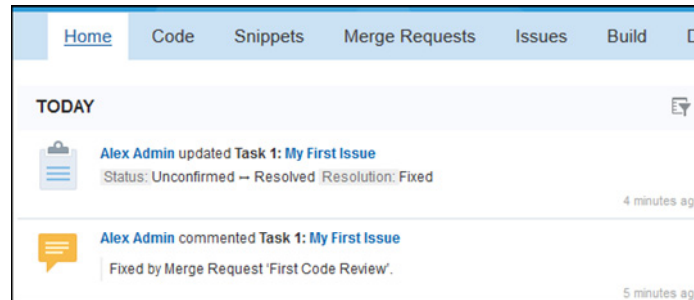


## Applications and Project Activities

- A project contains tabbed pages that organize your application.



- The Home page shows project activities.



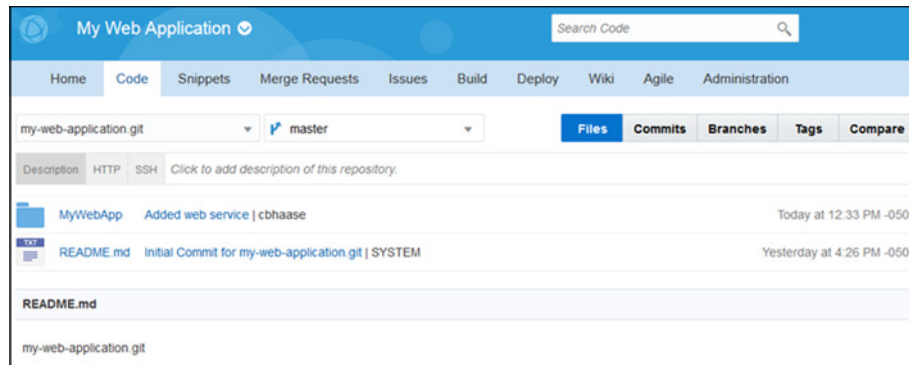
ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The project activities are on the left side of the Home page. The right side of the Home page has a news feed along with tabs that show the repository list, team members, and project statistics. Often, a project has many repositories and many team members.

## Repositories, Branches, Tags, and Commits

Use the Code page to see your source code repositories and branches, navigate down through the directory hierarchy, open files, and view their contents.



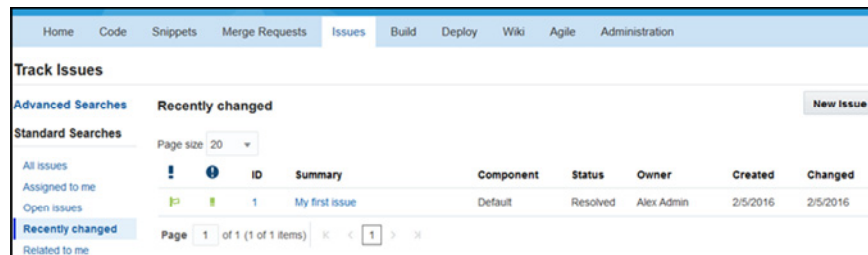
ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Typically, you store the project code on your local system and push it to the Cloud repository for building, reviewing, and deploying it. A large and complex project can have many repositories.

## Issues, Agile Boards, and Sprints

- Use the Issues page to create and manage issues.



- Use the Agile page to organize the issues into Agile boards and sprints.



ORACLE

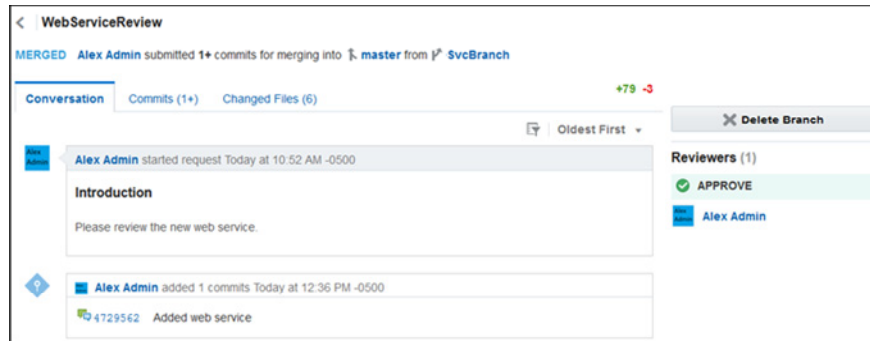
Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The main Issues page contains a list of issues. You can click an issue in the table to see details about the issue.

The Agile page has a Backlog view that lists the sprints and any issues that are not assigned to sprints. It also has an Active Sprints view where you can see the progress of the issues in the sprint.

## Code Reviews

Use the Merge Requests page to create and manage code reviews. The main page looks like the table on the Issues page. An individual code review page displays the history of the review.



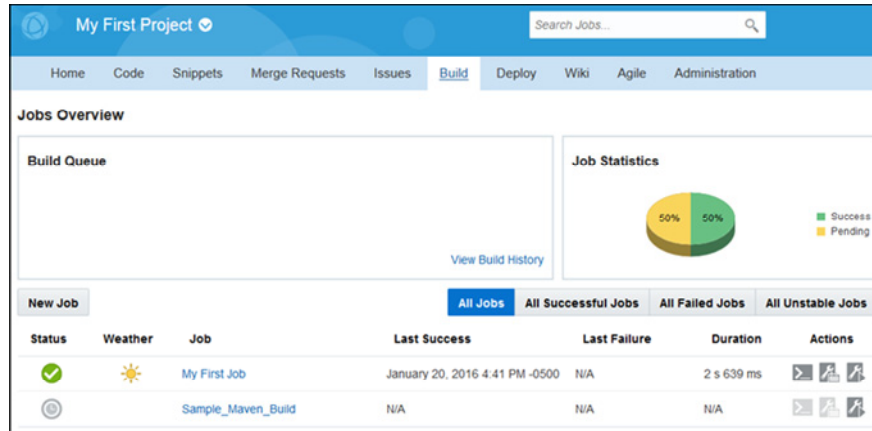
ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The main Merge Requests page looks much like the Issues page, with a list of merge requests in a table.

## Build and Deploy Artifacts

Use the Build page to manage builds for your application and then use the Deploy page to deploy artifacts that were built to Oracle Java Cloud Service.



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Other pages in Oracle Developer Cloud Service:

- The **Administration** page allows you to manage or delete project properties for projects you own.
- The **Wiki** page allows you to create wiki pages to document the project.
- The **Snippets** page allows you to create small snippets of text that you can reuse in wikis or descriptive text.

## How Do I Use Oracle Developer Cloud Service?

To work on an Oracle Developer Cloud Service project, perform the following steps:

1. Create the project. A Git repository is created for you.
2. Create issues to define the work that you will do on the project.
3. Set up an Agile board and sprints, and assign the issues to the sprints.
4. Clone the repository locally and work on the source code for a particular issue.
5. Build and run the project in the IDE to see if it works.
6. Create a merge request and check in the code to a branch.
7. When the code is approved, merge it to the master branch.
8. Close the issue.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Oracle Developer Cloud Service is meant for team development. Commonly, the issues that define the work on the project are assigned to members of the project team, who review each other's code for individual issues.

## Demo – Requesting for Oracle Cloud Trial Account

In this Demo you will learn to,

- Request and Activate Oracle Cloud Trial Account



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The MAVEN\_OPTS environment variable is used to specify JVM properties that can be used to supply extra options to Maven, such as the value -Xms256m -Xmx512m. Optionally, add this environment variable.

## Summary

In this lesson, you should have learned how to:

- Describe the concepts of the Scrum methodology
- Create user stories to describe requirements
- Estimate stories and team capacity
- Describe the fundamentals of test-driven development
- Create assertions in your code
- Design and create code based on tests
- Describe what Oracle Developer Cloud Service is
- List the main components of Oracle Developer Cloud Service
- Explain a typical workflow for using Oracle Developer Cloud Service



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.