

Continuous Deployment and Integration

Objectives

- To provide an overview of what deployment is
- To provide an overview of what integration means with respect to software development
- To provide a comparison between AGILE and traditional forms of the software development and delivery model
- Describe the key processes within continuous integration

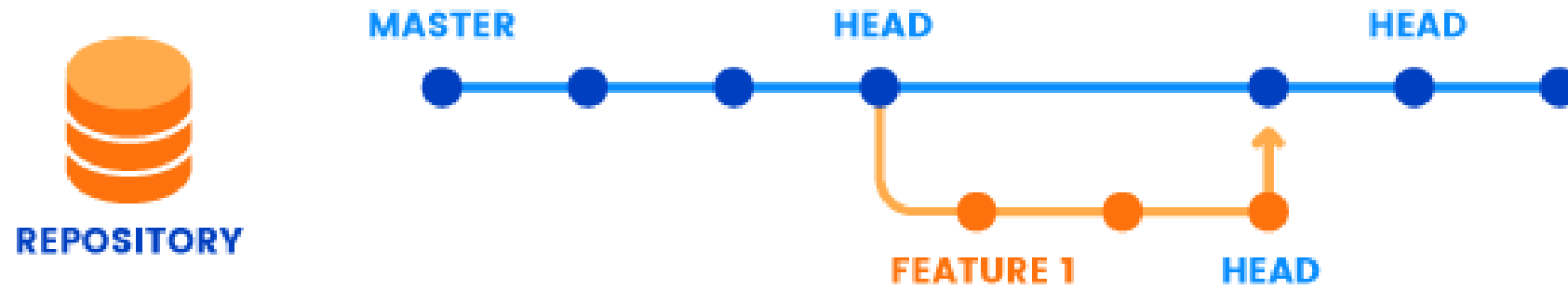
Development Teams recap

- AGILE software development is achieved utilising teams
- Development efforts include a manager a development team and a product owner (Particularly the case of SCRUM)
- Ensuring efficiency in such an environment can prove a large part of the undertaking
- To ensure the development team exhibits optimum efficiency what is utilised ?

Distributed software development

- Utilisation of a means of provisioning developers with individual copies of the source tree
- Allows for the developer to work with a local copy of the code form the main source tree and its associated branches
- Provides code metric attribution and performance information for each of the developers

Distributed Source Overview



Distributed Process

- After creation of a remote repository, the developer can create a local working copy of this repository
- Any changes that are made to this local repository are not known to the remote repository until it is made aware by the developer
- This means that the developer is able to create distinct individual branches within their local repository without having any adverse effects upon the main source tree of the project.

Remote awareness

- Once the developer has implemented the feature that they are required to for the project, they will need to make the main source tree aware of the changes.
- This is typically achieved by pushing the changes up to the remote repository from the local repository
- What happens if the code contains bugs ?
- What happens if the code contains malicious routines ?

A Testy Situation

- Typically, code is not merged into the source tree of a project directly before it has undergone extensive testing
- This includes a battery of:
 - Unit Testing
 - White box
 - Black box
 - Integration
 - Acceptance

Integration

- The process of merging the newly implemented code into the main source tree of the project is called integration.
- This process can result in a multitude of inconsistencies within the code variations between different developers, especially when a large number of contributors are working on the code.
- These inconsistencies are termed conflicts

Conflict

lib/compose.js

View file @e71062d

<pre>@@ -12,19 +12,27 @@ function setWritability(obj, writable) { 12 }); 13 } 14</pre>	<pre>@@ -12,19 +12,27 @@ function setWritability(obj, writable) { 12 }); 13 } 14</pre>
origin//their changes Use this <pre>15 function mixin(base, mixins) { 16 base.mixedIn = base.hasOwnProperty('mixedIn') ? base.mixedIn : [];</pre>	HEAD//our changes Use this <pre>15 function mixin(base, mixins) { 16 base.mixedIn = Object.prototype.hasOwnProperty.call(base, 'mi xedIn') ? base.mixedIn : [];</pre>
<pre>17 18 for (var i = 0; i < mixins.length; i++) { 19 if (base.mixedIn.indexOf(mixins[i]) == -1) { 29 module.exports = { 30 mixin: mixin 31 };</pre>	<pre>17 18 for (var i = 0; i < mixins.length; i++) { 19 if (base.mixedIn.indexOf(mixins[i]) == -1) { 20 setWritability(base, false); 21 mixins[i].call(base); 22 base.mixedIn.push(mixins[i]); 23 } 24 } 25 26 for (var i = 0; i < mixins.length; i++) { 27 if (base.mixedIn.indexOf(mixins[i]) == -1) { 37 module.exports = { 38 mixin: mixin 39 };</pre>

Conflict Resolution

- To mitigate the occurrence of this multitude of conflicts Grady Booch suggested the practice of continuous integration
- Up until this point, integration was seen as a mammoth task that was to be undertaken upon the completion of a software element.
- Booch suggested otherwise

Consider this scenario

- Your company has been contracted to create a piece of software that manages an MRI machine within a busy hospital in Swansea. The MRI machine should allow for a technician to initialise the machine with pre-defined parameters. Following this the technician should be able to adjust these parameters. A confirmation message should be displayed on screen to the technician upon attempting to start the machine.

Continuous Integration

- Booch suggested that to tackle the problem of what he termed “Integration hell” that the frequency of integrations should be increased reducing the amount of code that could present a conflict
- Considering the previous scenario, how would this benefit the development team ?

Conflict Resolution

- Conflicts are sometimes inevitable
- So how do we deal with them ?
 - In git, we use merge with associated options for handling some of the conflicts
 - Sometimes even git struggles with this and as such manual changes are required

↑ ↓ Highlight words

6 changes, 2 conflicts

Local Changes (Read-only)

LF Result

CRLF

Changes from Server (revision 50314cc8638a07fa4a88653d0752ec656f9... LF

```
public int getMin() {
    int min = data[0];
    for (int value : data) {
        if (value < min) {
            min = value;
        }
    }
    return min;
}

public int getAverage() {
    if (data.length == 0) {
        return 0;
    }

    int sum = 0;
    int i = 0;
    while (i < data.length) {
        sum += data[i];
        i++;
    }
    return sum / data.length;
}

public void printData() {
    // Print out the data
    for (int value : data) {
        System.out.print(value + ", ");
    }
    System.out.println();

    // Print out the stats:
    int avg = getAverage();
    int min = getMin();
```

```
* NumberFun class manages some data an generates
* My Changes and teammate's changes applied!
*/

public class NumberFun {
    private int[] data;
    public NumberFun(int[] data) {
        this.data = data;
    }

    public int getAverage() {
        int sum = 0;
        int i = 0;
        while (i < data.length) {
            sum += data[i];
            i++;
        }
        return sum / data.length;
    }

    public void printData() {
        // Print out the data
        int i = 0;
        while (i < data.length) {
            int value = data[i];
            System.out.print(value + ", ");
            i++;
        }
        System.out.println();

        // Print out the stats:
        int avg = getAverage();
        System.out.println("Stats:  avg = " + avg
    }
```

```
* NumberFun class manages some data an genera
* Teammate's changes applied!
*/

public class NumberFun {
    private int[] data;
    public NumberFun(int[] data) {
        this.data = data;
    }

    public int getAverage() {
        int sum = 0;
        for (int i = 0; i < data.length; i++)
            sum += data[i];
        return sum / data.length;
    }

    public int getMax() {
        int max = data[0];
        for (int value : data) {
            if (value > max) {
                max = value;
            }
        }
        return max;
    }

    public void printData() {
        // Print out the data
        for (int i = 0; i < data.length; i++)
            int value = data[i];
            System.out.print(value + ", ");
    }
```

Accept Left

Accept Right

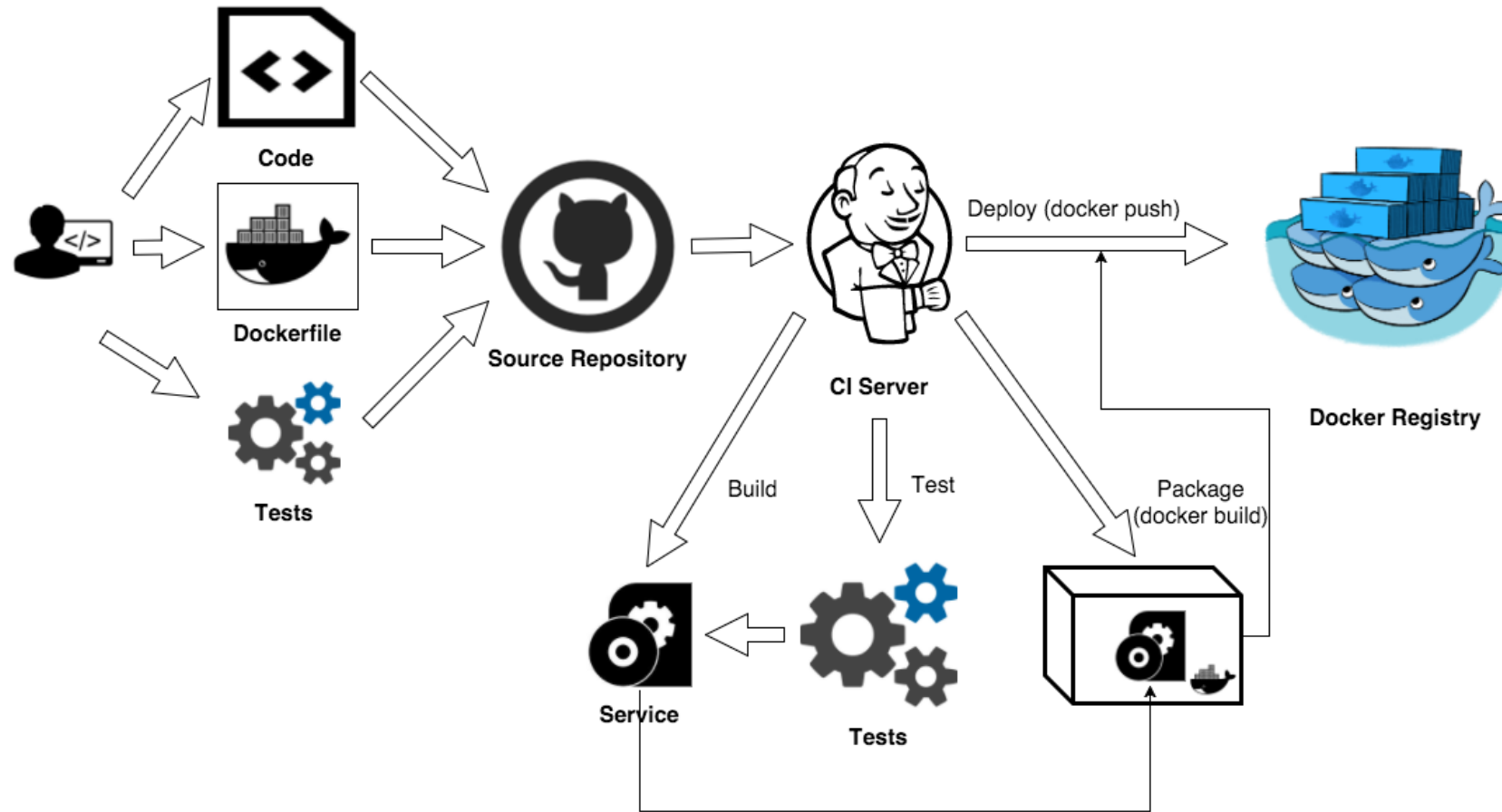
Apply

Abort

Continuous Integration - Process

- A developer will create a copy of the accepted code base to their working environment
- The developer will add the features and make the changes that they need
- The changes are then tested against any testing the developer needs to achieve then fed into the testing pipeline as determined by the development team and management
- This process is ongoing and may happen several times for the completion of a single feature

Continuous Integration - Pipeline



Test Automation

- Testing can be automated by software and as such much of the process involved in CI makes extensive use of this
- The development team will typically incorporate the use of a test suite written in house to initially test the changes
- Following this, the changes are then fed into a simulation environment as depicted by the previous pipeline overview

Test Suites

- A test suite is in its essence a collection of unit tests, integration tests and white and black box testing.
- This provides the stable test bed upon which the majority of functional testing is performed.
- This is usually a collaborative effort between the developers and QA team
- All new features are run against this to ensure the new features have not caused any issues.

Tools that aid in this process

- Jenkins - Management
- Docker – Simulated Testing
- Selenium – continuous testing
- Git – distributed development

Maginot Line

- Last line of defence against erroneous code
- Once all of the testing has been conducted on the new features the code will then be added to the main trunk
- Addition of code to the trunk is usually at the discretion of the development lead (squad lead)

Code Appended, What now

- What happens after a feature has been implemented and tested ?
 - It is then provided to the client / customer
- In traditional software development (utilisation of the waterfall method and incremental approaches) this is largely the end of the line for the development team with the exception of maintenance.
- How does this differ in AGILE software development

Where else can we apply this principle ?

- Remember the tag line for the approach
 - Frequent updates with limited volume
- This works well for the implementation of a feature
- What about the deployment of this feature ?

Software Deployment / Release

- Software deployment is the act of making the software available to the client.
- In traditional software methodologies this was an intensive task in which the entire system would be rolled out following some changeover method
- This large change in short spaces of time can cause a multitude of issues not least monetary expense

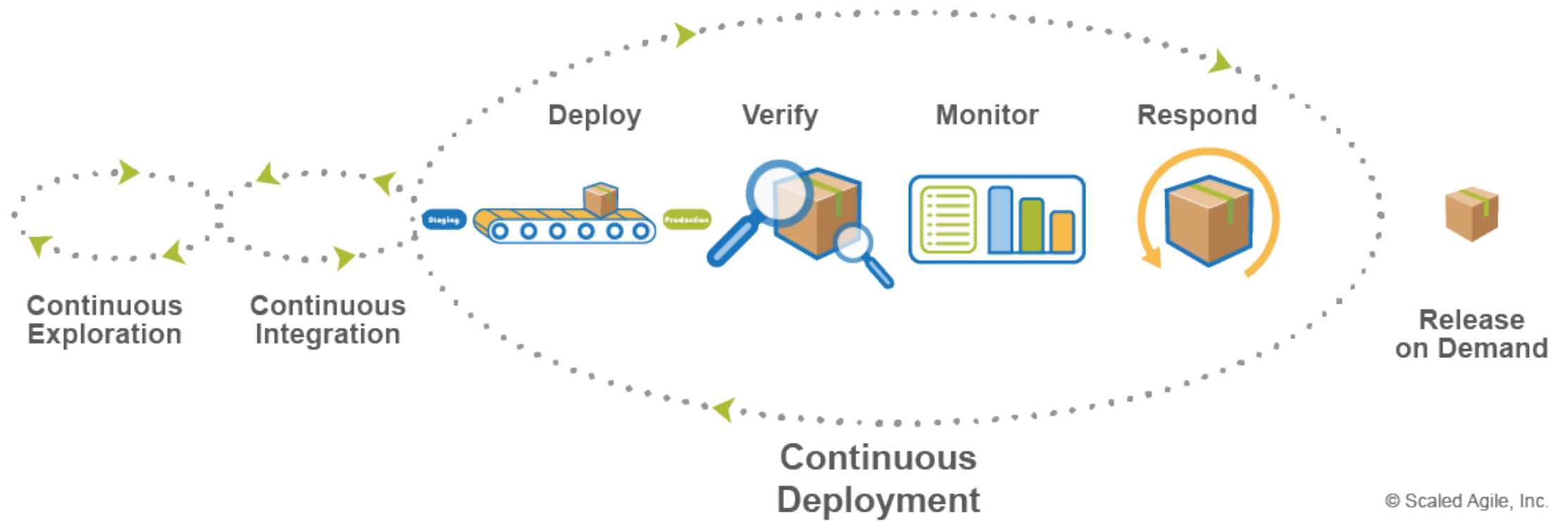
Reducing the intensity of deployment

- In much the same way as continuous integration the deployment process can be treated as a series of micro transactions or operations
- Following the increased frequency of continuous integration, CD provides the client with a much faster turnaround of features than that would be possible if the deployment was scheduled and staged in the same way as has been achieved traditionally

Continuous Deployment

- Every member of the team has visibility of the process at all times. This ensures that everyone is aware of the stage at which the project currently resides
- Adequate means of reporting is implemented within the software so that the development team is alerted to the occurrence of software faults, failures and bugs
- The changeover becomes far less daunting removing the mass panic involved. This means that developers have greater confidence that deployment of a new feature will go well

Process Overview



Advantages

- Reduces the size of an update
- As soon as a feature has been tested the client may make use of this
- Limits the errors that can occur within a change
- Shatters the ideology that a rollout is scary

Delivery Vs Deployment

- What is the difference ?
 - Continuous delivery follows the same process as continuous deployment, however, falls short of the deployment element.
 - The idea behind continuous delivery is that a feature has been implemented and tested and is proven to be functionally ready to ship
 - Continuous deployment enacts this last stage.

Plenary

- Continuous integration, adopts a greater frequency approach to the commit and merging of code to a repository
- Continuous integration doesn't necessarily stipulate a frequency however the ideology holds that even the smallest elements of a feature can be included
- Continuous deployment is a process by which the user is provisioned with a feature upon the completion of that feature. This increases the speed at which a user or client may utilise the software in contrast to having to wait on the completion of the entire software prior to its utilisation
- Continuous deployment in a similar fashion to integration, reduces the headaches for the developer with respect to required bug fixes in the event of errors.

Task

- Look at the features available by JIRA discuss as a group what process outlined they fall into. Do they provide adequate features for the facilitation of this process ?
- Looking at the features of Jenkins, does this provide the necessary features for the management of continuous delivery and deployment throughout a development cycle ?
- Locate another software element that provides for a means of aiding in this process. (You already know of at least one)