



Developing and deploying with Sitecore in four levels

Sitecore Development to Deployment Guide

This document will help you to define your development and deployment strategy, it describes how to set up your development, deployment, CM and CD environment as per the best practices which I've learnt while working on Sitecore since last more than 2 years!

Kiran Patil
Version 1.0
May 12, 2011

This document covers how you should set up your Development to Deployment Environment for Sitecore in your organization.

So less people think on this and they directly jump in to implementing Sitecore directly within their organization. This leads to be a complex life of everyone (Including Developer, Manager, and event Client as well!).

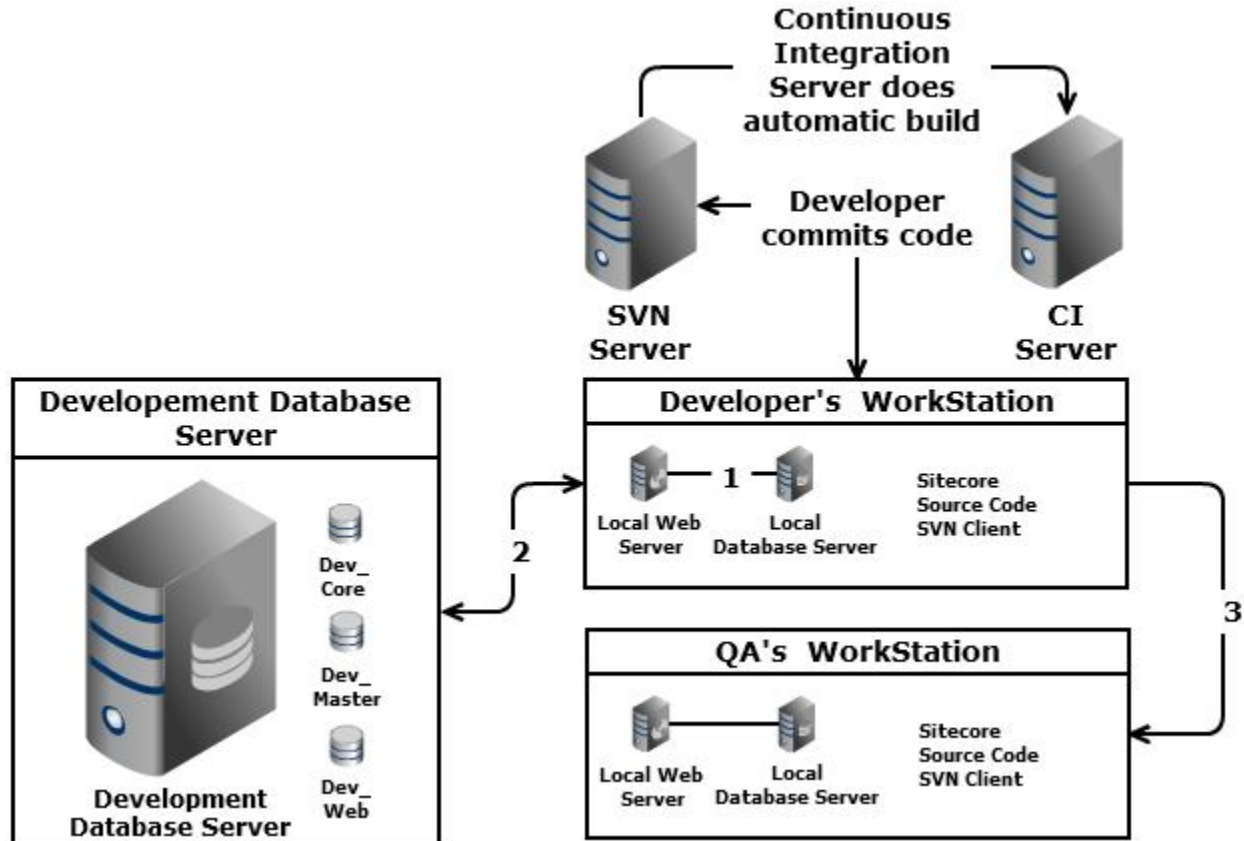
For simplicity I would like to slice each environment by Level. So, Sitecore development and deployment will have four levels as below:

- 1. Level1 (Team Development Environment)** – It will be referred as Team Development Environment. This is the first level and it focuses on Development and QA Environment. Basic but very important environment. If you can catch and fix your errors at this Level then your clients and your life will be smoother! (No weekend work!).
- 2. Level2 (Deployment Gateway Environment)** – It will be referred as Deployment Gateway Environment. This is the second level and it focuses on Integration deployment. This level will be bit identical to your live environments. For example if you have one CM and 10 CD Servers then here you should have 1 CM Server and X CD Servers. So, you can test Sub layout publishing, and all other things which works as per CM-CD concept (e.g. Event Queue and Web Deploy). Do your stress/load testing here. If you can catch all errors at this level then you never have unhappy clients and never you will see client's complaining that this thing stopped working which was working before deployment!
- 3. Level3 (CM Environment)** – It will be referred as CM environment. Too sensitive! As your CA, PE and CE users are working here. It needs some amount of down time if you are not using Multi Instance in CM environment.
- 4. Level4 (CD Environment)** – It will be referred as CD Environment. The most sensitive environment. Because your entire client facing front end websites will be served by this level. So, if error occurs here it will be LIVE Issue! No Downtime possible!

Let's go in details of each level.

Level1 (Team Development Environment)

We have already covered need of this environment in introduction section. Below diagram explains its overall working.



Before you start understanding above diagram, it will be worth to read some terminologies. Which you will be reading throughout this section:

Clean Database: - This database will be clean database and it will be shared across all developers. One should not play anything with this database and it will be restored from **Level2/3/4** environment. After that only Developer's can install packages in this DB. No R&D please!

Just a note: While using this database Sitecore Caching issue may come. Usually it won't as in Local Development environment developers regularly does Rebuild/Build of a solution which causes worker process to recycle and it clears the cache. In worst case if you face any caching issue do IISRESET/Recycle worker process.

Dirty Database: - This database will be always being dirty and it will be installed on each Developer's local machine. This database's sole purpose is for R&D. Do R&D on it as much as you can! Here developer is free to do any DB related changes!

Now, let's understand how it will be working.

We've three different environments in **Level1**; importance of each level is discussed below:

Development Database Server: - This will be the "clean database" in **Level1**. Here we will restore latest DB backup from **Level2/3/4**. After that whenever any developer works on any new functionality/BUG fix and does any change in Sitecore Database on his/her local machine, he/she should create package of it and install that package on this Database. And once done user should change his/her connection string and point it to this DB and Unit test the functionality with this DB again. We will be discussing the benefits of this approach in a while.

Just a note: It is responsibility of every developer to install package on this DB. This will make everyone's including his/her's life easy! Most important thing is that before installing any packages on this DB please make sure your package is packaged in a right way! (Have any confusion? Better to ask your peer to review your package).

Hardware & software requirements: This machine should have good disk space and should have installed SQL SERVER 2008.

Developer's WorkStation: - Here developers will be working. They do have "dirty" database's local copy here they can do all R&D, Main development and Bug Fix. Once the task is completed Developer should change his/her connection string and point to "clean" database and do the final unit testing before sending to QA.

Just a note: It's always best practice to update your solution from SVN after each deployment.

Hardware & software requirements: Developer's machine with Visual Studio, SVN, Sitecore, IIS and SQL SERVER needs to be installed.

QA's WorkStation: - QA will have their local DB copy with local sitecore and application connected via SVN. Here they will be deploying package and code sent by Developer. Developer should not access this Database for anything like Add/Modify/Update anything.

QA will get package (if exist) and code files from developer. They will install package on local machine and then proceed for their QA process.

Just a note: QA's local copy should be up to date with SVN repository. It's always best practice to update your solution from SVN after each deployment. QA should ensure that developer has Unit tested the functionality properly and installed package on "clean" database before starting their testing.

Hardware & software requirements: QA's machine with Visual Studio, SVN, Sitecore, IIS and SQL SERVER needs to be installed.

SVN Server: - Once QA gives green signal to developer that he/she is happy with the changes. Then developer should check in the code in SVN along with package as well (If exist). It will act as a central code repository system.

Just a note: It's always best practice to update your solution/file before you commit any code, and also do compare and ensure that your code changes are not being conflicted with your peers! If so please do proper merging before commit. Also, please write proper comment before your commit! It will make everyone's lives easy!

Hardware & software requirements: SVN Server might be on LINUX/Windows. Whatever you wish to use!

CI Server: - Continuous Integration plays a vital role in Agile Development. It will do automatic build on preconfigured time. Once Build done you can copy the required DLL, Files (aspx, xml etc.) and deployment packages and create Deployment package for **Level2**. It will act as a Continuous Integration system. If Build gets failed/successful it should send mail to everyone and notify that build was failed/successful!

Hardware & software requirements: Machine with Visual Studio, SVN, IIS and SQL SERVER [Optional], Nunit [Optional] If you have some unit tests running on commit, Continuous Integration software (CruiseControl.NET/NANT – Personally I recommend CruiseControl.NET!). Integrate CI Server with IRC channel and Email System.

This is how it works step by step:

1. Whenever developer gets new task. He/She will be doing the necessary code/database changes in local environment where he /she have local DB copy. If developer does any database change then he/she should create package of it and then move to step2.

Below are some benefits of following this step:

- As every developer will have his/here "dirty" copy they can do R&D on it without taking anyone's permission and affecting others work.

2. Once developer is done with Step1, he/she should change local copies connection string and point it to "clean" database, install it on "clean" DB and do proper Unit Testing before going to step3. Please note if developer has not changed anything in DB then also he/she should point to "clean" DB and do proper unit testing.

Below are some benefits of following this step:

- As every Developer must need to install package on "clean" DB, we will have all DB changes (e.g. Template/Field change) at one centralized DB.
- It will be good for doing partial Integration Testing (Only DB Changes) for every developer.
- Regression testing will also be covered while doing Unit testing.
- If anyone would like to check anyone's functionality after deployment, then he/she should take Latest from SVN and point his/her local copy to "clean" DB. No Package search and hassle of configuring the scenario!

- Once developer is done with Step2, he/she should send updates (Code/Package) to QA. Here QAs will be configuring the updates locally and doing QA.

Below are some benefits of following this step:

- As developer has tested everything in Step2. There will be less chance to have errors in this level. (We did Regression and Integration in step2!).
- It will be good for doing full Integration Testing (DB + Code Changes) for every QA.

Just a note: If QA raises any BUG then Developer should follow Step1 to Step2 before moving to Step3.

- Once QA gives green signal to developer then he/she should check in the code in SVN.

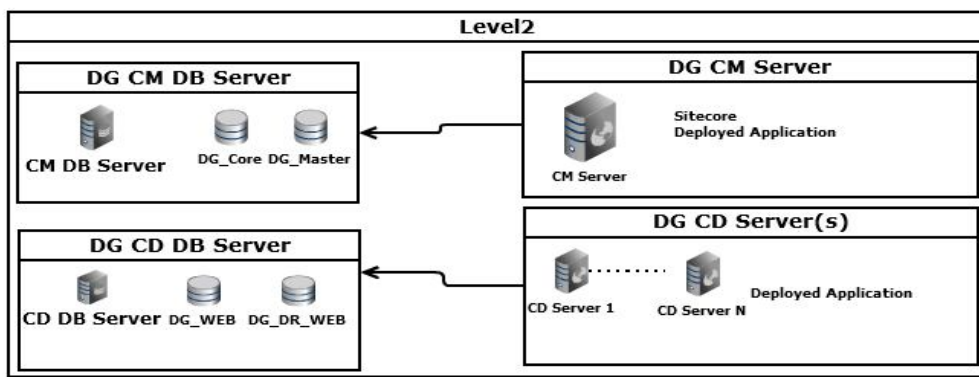
Just a note: You should commit all your packages as well in repository. So, whenever someone would like to install any DB change in his/her "dirty" DB he/she should be able to do it after taking package's update from SVN.

- Once CI Server sends Build Successful notification. QA guys should proceed further with creating Deployment package for **Level2**.

We should move the Deployment to **Level2**, if any only if we have followed **Level1**.

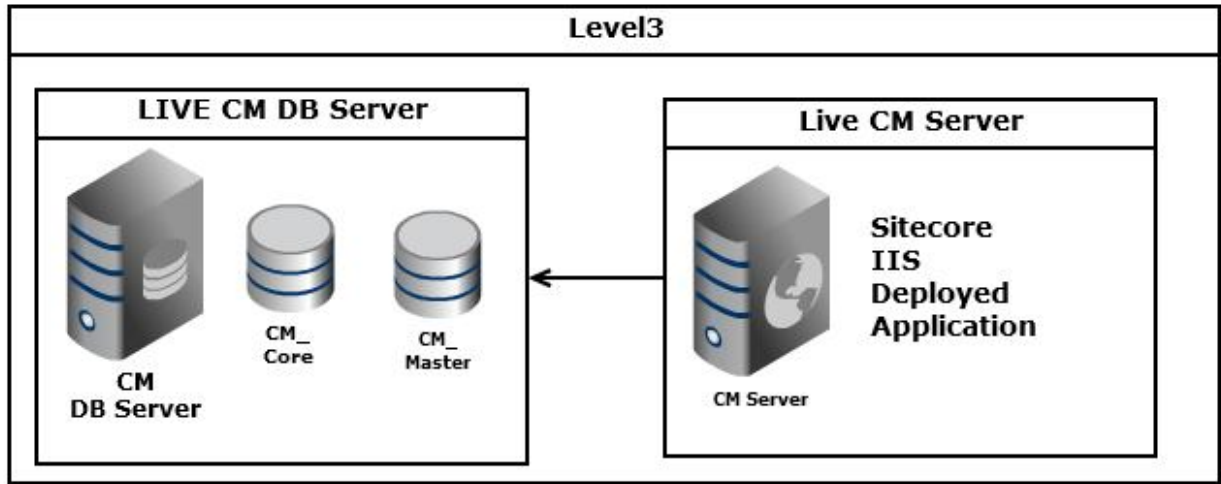
Level2 (Deployment Gateway Environment)

This environment will act as a deployment gateway environment. We have already covered need of this environment in introduction section.



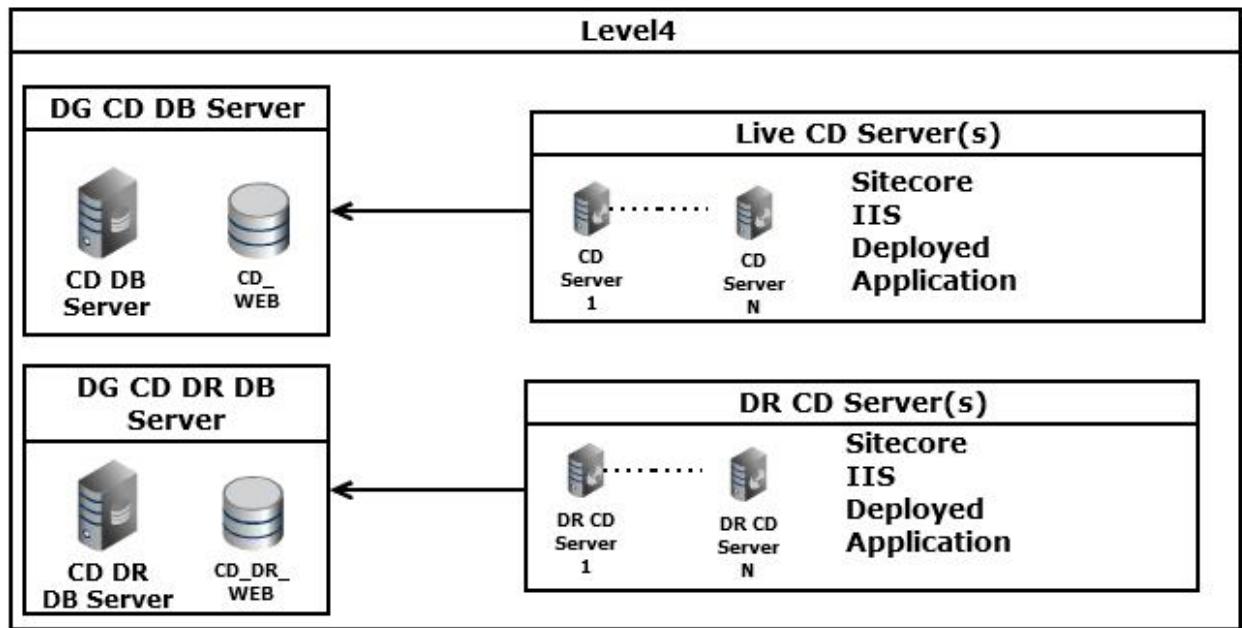
Level3 (CM Environment)

This environment will act as a content management environment. We have already covered need of this environment in introduction section. You can also consider this as your Pre-Live environment!



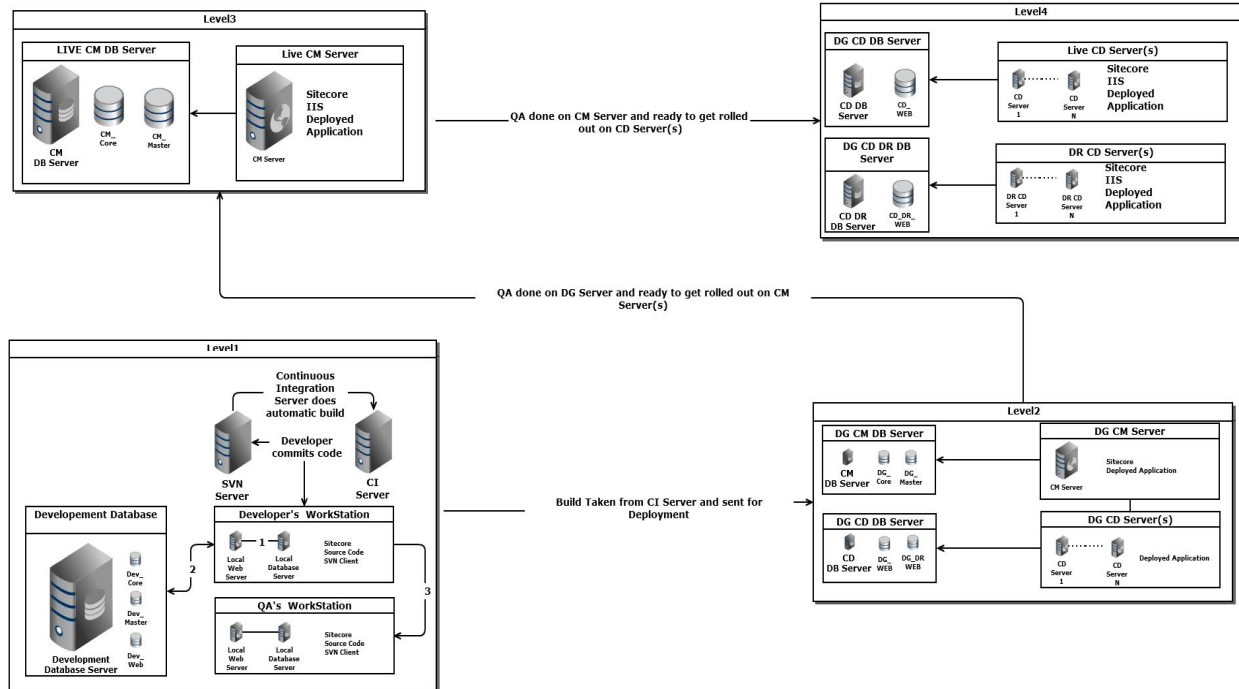
Level4 (CD Environment)

This environment will act as a content delivery environment. We have already covered need of this environment in introduction section.



From Level1 (Development) to Level4 (Deployment) environment at a glance

Below diagram shows flow of each Level.



Happy Sitecore Development to Deployment! ☺

Webliogrphay

Link	Description
http://sitecorebasics.wordpress.com/2011/04/25/basics-of-sitecore-hosting-architecture/	Sitecore Hosting Architecture
http://sitecorebasics.wordpress.com/2011/05/01/team-development-with-sitecore/	Team Development with Sitecore