Server Migration Checklist

By Andrew Hinkle

The goal of this whitepaper is to provide a guide to assist you in identifying what needs to be migrated. It does not necessarily provide steps on how to perform the migration. In many cases you'll have enough information to deduce those steps on your own and the rest can be searched on the web.

As a Full-Stack developer you are an integral piece in the effort to successfully migrate the company's applications to the latest and greatest server. All too often, developers assume the operations team is performing a task and vice versa. In the age of DevOps it’s the responsibility of both to work together. As a good friend of mine points out the Dev in DevOps includes QA, probably because DevQaOps doesn't roll off the tongue as smoothly.

A developer should bring to the discussion the following basic server migration checklist with reminders. These responsibilities should be delegated between appropriate personnel. As always "appropriate" is dependent upon your team's culture, skill level, and patience. This process can be frustrating, tedious, annoying, and nightmare inducing. Not that I have experienced any of that. Ahem, let's continue.

The examples for each step are based on Windows servers and while some navigation steps are provided every OS has its own quirks, but at least you have been guided in the correct direction. The same concepts may be applied to other operating systems. Keep in mind, this is the list you may start with and expand as you learn more about your systems. Please reply with any enhancements to this list.

Server Migration Checklist

[Team 5](#_Toc490984637)

[Developers, QA, Operations, DBA, Reports, Business 5](#_Toc490984638)

[Meet regularly, track progress, and always have a list of action items at the end of each meeting 5](#_Toc490984639)

[Migration options 6](#_Toc490984640)

[Server Alias 6](#_Toc490984641)

[Naming convention 6](#_Toc490984642)

[Domain 7](#_Toc490984643)

[Clean up 7](#_Toc490984644)

[Windows Update 7](#_Toc490984645)

[Migration to the Cloud 7](#_Toc490984646)

[Hardware Upgrade 7](#_Toc490984647)

[Operating System Update 8](#_Toc490984648)

[Documentation 8](#_Toc490984649)

[Existing Documentation 8](#_Toc490984650)

[Document 8](#_Toc490984651)

[Credentials 9](#_Toc490984652)

[Environments 9](#_Toc490984653)

[LOCAL 9](#_Toc490984654)

[DEV 9](#_Toc490984655)

[SIT 9](#_Toc490984656)

[UAT 9](#_Toc490984657)

[PROD 9](#_Toc490984658)

[Example differences between environments 9](#_Toc490984659)

[Users and Groups 10](#_Toc490984660)

[Permissions 10](#_Toc490984661)

[Global Policy 10](#_Toc490984662)

[Permissions to Log on as a service and batch job 10](#_Toc490984663)

[Component Services 11](#_Toc490984664)

[Search for references to server name 11](#_Toc490984665)

[Scheduled Tasks 11](#_Toc490984666)

[Event Viewer 11](#_Toc490984667)

[Shared Folders 11](#_Toc490984668)

[Services 11](#_Toc490984669)

[ODBC 12](#_Toc490984670)

[Web Hosting – IIS/Apache/etc. 12](#_Toc490984671)

[Application Pools 12](#_Toc490984672)

[Sites, Applications, and Virtual Directories 12](#_Toc490984673)

[Web.configs 13](#_Toc490984674)

[Source Control 13](#_Toc490984675)

[Deployment Process 15](#_Toc490984676)

[InstallShield, Batch, PowerShell, etc. 15](#_Toc490984677)

[Build/Release Definitions 16](#_Toc490984678)

[Continuous Integration (Build Definitions) 16](#_Toc490984679)

[Continuous Deployment (Release Definitions) 16](#_Toc490984680)

[Database 17](#_Toc490984681)

[Search for references to server name 17](#_Toc490984682)

[Database Migration 17](#_Toc490984683)

[Database is stored on this server 17](#_Toc490984684)

[Database Connection Strings 17](#_Toc490984685)

[Reports 18](#_Toc490984686)

[Applications 18](#_Toc490984687)

[Add/Remove Programs 18](#_Toc490984688)

[Programs Files 18](#_Toc490984689)

[Miscellaneous Folders/Files 18](#_Toc490984690)

[Registry 18](#_Toc490984691)

[Folders 18](#_Toc490984692)

[Miscellaneous Folders/Files 18](#_Toc490984693)

[Dependencies 19](#_Toc490984694)

[Security 19](#_Toc490984695)

[Anti-virus 19](#_Toc490984696)

[Firewalls 19](#_Toc490984697)

[Whitelists and Blacklists 19](#_Toc490984698)

[Email 20](#_Toc490984699)

[Cleanup 20](#_Toc490984700)

[Rollback Plan 21](#_Toc490984701)

[Developer Testing 22](#_Toc490984702)

[Communication 22](#_Toc490984703)

[Identify how every effected application and their dependencies are used 22](#_Toc490984704)

[Turn off the old server 22](#_Toc490984705)

[QA Testing 22](#_Toc490984706)

[Create test plans 22](#_Toc490984707)

[Update manual and automated regressions tests 23](#_Toc490984708)

[DEV/SIT/UAT/PROD 23](#_Toc490984709)

[DEV 23](#_Toc490984710)

[SIT 23](#_Toc490984711)

[UAT 23](#_Toc490984712)

[PROD 23](#_Toc490984713)

[External (public facing) 24](#_Toc490984714)

[Conclusion 24](#_Toc490984715)

# Team

The scope of the team depends on what is on the servers and who they serve.

## Developers, QA, Operations, DBA, Reports, Business

1. Developers are tasked with updating server name references in source control and databases.
2. Developers and Operations work together to identify the company and third party applications and most of the minutia in this article.
3. Quality Assurance (QA) performs automated and manual regression tests to ensure there is no lost in functionality.
4. Operations setup the new server and install all of the company applications from SIT to PROD and third party applications from DEV to PROD.
5. DBA will setup/migrate the databases, sys.types, sys.messages, users and roles, etc.
6. DBA and Reports work together to setup Data Sources
7. Reports team migrate their reports.
8. Business is kept in the loop regarding the progress and can verify application usage by reaching out to the users.

## Meet regularly, track progress, and always have a list of action items at the end of each meeting

1. Have a kick off meeting to get everyone involved on the same page.
2. Every meeting should include a review of a spreadsheet of action items stored in a centralized location everyone can access such as SharePoint or a Shared Folder.
   1. User: The person to perform the action. Depending on size this may be by team instead of individual.
   2. Action: Description of the action.
   3. Notes: Any additional notes to help the user. Perhaps it can't be worked on until another task is done.
   4. Roadblock: Anything preventing the action from progressing once it is in progress. Perhaps a company application was found to be in use and the developer must now track down the installation.
   5. Status: Can be as simple as empty (todo), in progress, done (consider date completed).
3. Every meeting should include an agreement on when to meet next.
   1. You may start once a week for a while until enough analysis is complete
   2. Once environments start getting updated, you should consider meeting more than once a week depending on how quickly it is moving.
      1. Delay too long and you may be considering updating to the next version of an operating system or SQL Server before you even get through half the environments.
4. When finished or possibly at different milestones such as successful migration of an environment, have a retrospective with the goal on process improvement not blame.
   1. What worked?
   2. What could be done better?
   3. What didn't work?

# Migration options

This document revolves mostly on information gathering, troubleshooting, and testing, leaving the actual server setup to Operations, so this document is relevant regardless of which type of migration you perform.

So many types of migrations, I can't name them all, but here are some. Migrations typically start with the need for a hardware upgrade and then someone notes that we'll have to update the operating system, because the previous one is no longer supported and riddled with security holes. Suddenly, someone decides to clean up the system, because there are several applications on it that we no longer support. Regardless, the scope increases and task becomes monstrous and fragile.

While you may consider performing multiple types of migrations at the same time, I recommend that you isolate the migration steps as much as possible. Most times the hardware upgrade pretty much demands the operating system upgrade, but the rest can be mitigated. Yes, it will take longer. Yes, there will be more tests. Yes, there will be more work. Yes, you'll have a higher chance of success and less to consider when something goes wrong.

## Server Alias

Goal: Applications are using your server by the actual server name. Instead, you create a server alias that references the IP Address of the actual server name. This is very involved and intrusive as many connection strings and references now must be updated. You are considering this option, because you have to create a new server and you aren't using an alias already.

Options: If you are creating a new server, then you could make an alias with the same name as the server, so no changes have to be done. Once the new server is up and ready, change the alias to the new server's IP address and shutdown the old server and run tests.

### Naming convention

Of course if your servers were named after Pokémon, Egyptian gods, or past girlfriends, you may want to come up with a better alias name. Here's an example I've seen many times. Plenty of standards out there, so find one your company likes.

1. Alias: {company initials}-{product}-{environment}
   1. dw-danylkoweb-dev
   2. dw-danylkoweb-sit
   3. dw-danylkoweb-uat
   4. dw-danylkoweb-prod
2. Server: {company initials}-{product}-{environment}-{unique server number}
   1. dw-danylkoweb-dev-4
   2. dw-danylkoweb-sit-2
   3. dw-danylkoweb-uat-1
   4. dw-danylkoweb-prod-1

## Domain

Goal: You need to migrate the server from one domain to another. At this point you are more interested in searching for the specific domain name instead of server name. Keep in mind that the domain will be associated with the server and users such as for connection strings.

When considering domain migrations be very wary of the servers it depends on and depends on it. These other servers may need to migrate at the same time. Perhaps consider removing or isolating the dependencies as the first pass before the spider web of servers grows significantly making this a monumental all or nothing endeavor.

Keep in mind that if you are changing domains, then the users need to migrate to the new domain as well. Network Administrators may need to setup conditional forwards and setup the users with two-way Active Directory Trust. They may setup the users on the new domain to only trust the new domain to help verify during testing that the old domain is no longer referenced.

## Clean up

Goal: Delete applications no longer used by the company. Remove test applications. Simplify processes. Keep dependencies in mind and track down usage before removal. Loose coupling principles apply to Infrastructure just as much as coding.

## Windows Update

Goal: Windows Updates have been reviewed and approved. They will be installed through the environments, so QA may run regression tests to verify there is no lost in functionality. These are usually straight forward. Keep dependencies in mind as some Windows Updates as example have turned services off that may be required.

## Migration to the Cloud

Goal: You've decided to migrate the server to the Cloud. Keeping with the Microsoft theme, review the Microsoft Azure cloud and determine what path is best for you.

1. [Software as a service (SaaS)](https://azure.microsoft.com/en-us/overview/what-is-saas/)
   1. Hosted applications, plus everything in PaaS.
2. [Platform as a service (PaaS)](https://azure.microsoft.com/en-us/overview/what-is-paas/)
   1. Development tools, database management, business analytics, plus everything in IaaS.
3. [Infrastructure as a service (IaaS)](https://azure.microsoft.com/en-us/overview/what-is-iaas/)
   1. Servers and storage, networking firewalls/security, data center physical plant/building.

## Hardware Upgrade

Goal: Create a new server that performs all the same functionality as the original server with the intention of replacing the original.

As an example, you are creating a new server with better hardware, perhaps migrating to VDI or another type of Infrastructure as a Service that mirrors the environment. Regardless, you couldn't clone the server and had to setup from scratch or with incomplete templates.

## Operating System Update

Goal: Upgrade the operating system (OS) and ensure all applications still work knowing that some company and third party applications may need updated to work again.

If you are upgrading the operating system, you should highly consider creating a new server first. Whether it's a clone that you upgrade the OS or a fresh server with just the new OS, never perform this task on the original server. Too much can go wrong.

If you are not using a server alias, consider creating one and implementing it first. This way all code changes revolving around server name changes is performed independently of any changes required to make the applications work in the new OS.

# Documentation

## Existing Documentation

1. Hahahahahahahahahaha <dramatic pause>…</dramatic pause> hahahahaha, yeah right. Don't get me started. I have very rarely been pleasantly surprised to ever find any existing documentation. I typically come in with a clean slate and am tasked to "figure it out".

## Document

1. Do it.
2. Use this article as a guideline to outline all of the dependencies of the server and what is dependent on it. Then you can use it like a table of contents to reference proper installation/repair/maintenance instructions. Do it.
3. Are you seriously considering hoarding this information for "job security"? I wouldn't want to work with someone who does that. You know who I want to work with? Someone who documents properly. Do it.
4. A wise person once told me that in order to properly be noticed in a team or company is to be the one who does the action items others dread or refuse to do. No, I'm not talking about getting coffee. I'm talking about documentation. Do it.
5. Don't have time to document? When something critical goes wrong the week of deploying the production server what will you refer to recall what you did to resolve the issue in the QA environment? Do it.
6. Don’t have the patience? Will you, your colleagues, other teams, management, stake holders, and clients have the patience to wait for you to rediscover what you did to fix the issue the first time? Do it.
7. Feeling the subliminal message residing through this section of the article? Pretty long and annoying? Maybe I've been burned a little about the lack of any documentation that might have saved me an hour, day, or week to track an issue down? Maybe I just like to ask a lot of rhetorical questions? Do it.
8. Do it.

# Credentials

When identifying user's throughout the list, ensure you include the domain.

Hopefully someone on your team has KeePass or another tool for securely storing credentials. If you are using Excel, even if it is encrypted, then shame on you. Get a more secure modern app to store your passwords. Sadly, if you are storing your password on Post-It notes, then I've already got it. "1-2-3-4-5. it's amazing, I've got the same combination on my luggage."

# Environments

Enterprise level environments typically have five types of environments. These environments may or may not be on the same server. The closer you can keep each environment including hardware and software (company and third party applications) the same as production, the best.

Testing against these environments is discussed at the end of the article.

## LOCAL

This is the developer's local machines.

## DEV

Developers may be able to spin up new instances that mimic production enough for them to test their code. This may be a server that mimics production that developers share.

## SIT

Quality Assurance (QA) team perform there tests in the System Integration Test (SIT) environment.

## UAT

This is a subset of actual users test in the User Acceptance Test (UAT) environment.

## PROD

This is the production server or client machines.

## Example differences between environments

Here are some situations that arise when you're environments are not the same:

1. Hardware was not the same between environments. While tracking down a performance issue, unit tests demonstrated great improvements, however, actually running the tests against the changes in the application on the LOCAL and DEV environments showed some benefit but not drastic. Deployed the changes to the SIT environment which was closest to PROD and the performance enhancements were obvious.
2. A company application installed in only the DEV environment and wasn't a new project caught in the middle of the migration. Cool, don't migrate.
3. A company application installed in the DEV, SIT, and PROD. Why wasn't it on UAT? Previous developers figured it was a waste of time, because only administrators used it, so users couldn't test it in UAT. Sigh, QA can still test it, so I set it up.
4. A company application installed in the DEV, UAT, and PROD environment. Why wasn't it on SIT? SIT environment was added later and the application was missed during setup. It happens, I set it up properly.
5. A company application installed in the DEV and PROD environment. Why wasn't it on SIT and UAT? Started to get scared here. Thankfully, it was a dead application, but never learned why it didn't exist in the others.
6. A company application installed in the PROD environment only. Panic stricken I discovered that the application was dependent on a third party application that we only had the one license for Production. The company wasn't willing to purchase another license. Learned that the app only read data from the third party web service call to update a database by environment. I setup the application in each environment and worked with QA to develop tests.
7. A third party application that developers add scripts to with a UAT and PROD environment. I'm not sure if licensing was involved, however, I was told that setting up the UAT environment was very involved and time consuming. Enough so, that it was only updated once every three months to a year as necessary. Scripts were typically updated directly in production in a test area. QA struggled with testing the stored procedure I updated that the scripts called due to the environment being exact, out of date database, and no experience of the third party application. I still shiver every time I remember this experience.

# Users and Groups

## Permissions

1. Identify local users.
2. Identify the users and domain associated with each group.

# Global Policy

## Permissions to Log on as a service and batch job

1. gpedit.msc
2. Expand Computer Configuration > Windows Settings > Security Settings > Local Policies > User Rights Assignment
   1. Log on as a service
   2. Log on as a batch job

# Component Services

In order to support communications between VB6 and C# (and many other reasons) Component Services bridges the gap between COM components.

## Search for references to server name

1. Open "Component Services" and expand Computers > My Computer > COM+ Applications > Right-click each and click Properties
   1. Activation tab has the server name
   2. Identity tab has the user authentication

# Scheduled Tasks

1. Open "Computer Management" > "System Tools" > "Task Scheduler" > "Task Scheduler Library".
2. For each scheduled task > Right-click > Click Properties
   1. General: You're looking for what user the task is running as and if the scheduled task is to "run whether user is logged on or not"
      1. The user must have the Global Policy permission to "log on as a batch job".
   2. Triggers: Tells you when the scheduled task should run. If none of them are enabled and it is a company application, consider removing it.
   3. Actions: Informs you of the application that will run.
   4. History: When the scheduled task was last run
3. With this information you should be able to make an informed decision as to whether the:
   1. Scheduled task for the company application should be migrated.
   2. The third party application should be installed.

# Event Viewer

1. Open "Computer Management" > "System Tools" > "Event Viewer" > "Windows Logs" > "Application"
2. Might was well look through the logs and gauge what is being used or throwing exceptions. You'll probably visit this again later once you get further along with your analysis.

# Shared Folders

1. Open "Computer Management" and identify the "Shared Folders".
2. Open the properties for each shared folder and review the security permissions.

# Services

1. Open "Computer Management" > "Services and Applications" > "Services".
2. There are MANY services, so prioritize by first sorting by the "Log On As" column to identify the services run by users created by the company usually identified by {domain}\{user}.
   1. This user must have the Global Policy permission to "log on as service".
3. Next sort by Name and skim through the list to see what they are. You should be able to gauge them by names including the company name, or third party applications you are probably running.
4. Identify the status of each company service and third party service you are considering migrating as it may give a clue if the application or service is even being used.

# ODBC

1. Open "Control Panel" > "System and Security" > "Administrative Tools" > "ODBC Data Sources"
2. Under the "User DSN", "System DSN", and "File DSN" tabs > For each Data Source entry click "Configure"
3. Look through all of the settings available to see if there are any references to the server name.
4. Give the remaining tabs a cursory glance to see if there are any references to the server name.

# Web Hosting – IIS/Apache/etc.

1. Open "Computer Management" > "Services and Applications" > "Internet Information Services (IIS) Manager".

## Application Pools

1. Open the Application Pools and identify the entries. The Identity is very important as you'll need to recreate the company Application Pools and reenter the credentials.

## Sites, Applications, and Virtual Directories

Sites are identified with globe icons. Applications are identified with globe icons with several files on the bottom left. Virtual Directories are identified as shortcut folders. Icons vary between IIS versions, but this gives you a frame of reference.

1. For each web site
   1. Bindings identify the different IIS protocols used to connect to the sites, such as http and https, host name, port, and IP address. Also identify the SSL certificates for the https entries.
2. For each site, application, and virtual directory
   1. Research how each are installed.
   2. Edit permissions navigates to the application's folder permissions and identify all of the groups and users and their permissions.
   3. Basic Settings identify the Application Pool and Physical path.
      1. Connect As and identify if it uses "Application user (pass-through authentication)" or if a specific user is used. If Application user is used, then the user associated with the Application Pool is used.
   4. Advanced Settings identify provides the same info as above though you should make sure these settings are the same.

## Web.configs

I reserve the search for references to the server name when working in the Source Control and the Deployment Process.

1. In this section, you are concerned with identifying any web.configs that are on the server, actively used, and not found in Source Control. Find the web.config files and verify they are in Source Control. If they are not, check them in.
2. Identify web.config inheritance
   1. Web.config files are comparable to Apache .htaccess files and as such they share the ability to inherit from their parents.
   2. Sometimes developers take on the DRY (Do not Repeat Yourself) principles too far with configurations by creating a web.config in the site that contain database or other server references used by the applications under the site.
   3. Instead, practice separation of concerns with configuration files. The site or application currently resides on the server, however, in the future servers may be consolidated or split and it's easier to migrate them if there are no dependencies.

# Source Control

With a Microsoft shop Team Foundation Services (TFS) is the go to Source Control, though GitHub is starting to eat the world. I'll speak on using Visual Studio with TFS on premise in this section, though the principles are sound, just use whatever tool or feature that makes the most since for your repository of code.

I don't know of any easy way to quickly search for strings inside your files. There is an extension if you are looking for file names, but that won't help here.

1. Sanitize your local environment before continuing.
   1. Suspend (My Work), Check-in, shelve, or undo appropriately until you have nothing in your Pending Changes
   2. Delete local folders and files in your local copy of the Project Collections. You could also rename your local copy of the Project Collection so you may just delete the newly retrieved project collection after your search. You may also decide to just continue and get latest in the next step and deal with having all the excess files on your PC. Whatever you want, but don't yell at me if you lose something because you didn't have it bound to Source Control such as test results you left in the folder and forgot were in there.
2. Identify the main/trunk branch
   1. A branch contains all of the source code required to build/compile your code and typically contains the Solution .sln file with project folders or perhaps one folder up so the code is in a "src" folder, documents in a "docs" folder, etc. Look up branches if you're confused by any of these concepts.
   2. These are the files you expect to be in or going into production
3. For each Project Collection > navigate to the trunk branch > right-click on any folder that you don't want to download and click Cloak.
   1. Don't Cloak Database folders as you may catch some SQL references missed when you search the database later.
   2. Folders that contain static files like PDFs.
   3. Be careful with what you Cloak. You never know what crazy and "ingenious" things are forbearers were forced to do under great duress in the past. After all, you weren't in their shoes, so don't judge.
4. For each Project Collection > navigate to the trunk branch > right-click the trunk > Advanced > Get Specific Version, check the two Overwrite files > click Get.
   1. Depending on how much you have to download, you may need to perform your search one Project Collection at a time or even a couple dozen applications at a time.
5. Once you have all of the files downloaded > Click Edit > Find and Replace > Find in Files (Ctrl+Shift+F). Usually you are looking for a literal string, so don't get fancy here.
   1. Find what: {server name}
   2. Look in: {path to folder that contains your project collections}
   3. Find options:
      1. Uncheck Match case
      2. Uncheck Match whole word
      3. Uncheck Use Regular Expressions
      4. Look at these file types: \*.\*;-(\*.pdf)
         1. This filter looks at all file types: \*.\*
         2. Separator: ;
         3. Exclude PDFs: -(\*.pdf)
   4. Click Find All
6. Copy the results (file names with line numbers) into Excel (each entry should be a row) > select the first cell > select all (Ctrl+A) > Home: Format as Table > Choose a table (I like the Blue table on the first Medium row) > Click OK > Rename the column to "Search Results".
   1. Each server name you search should have its own Excel Sheet allowing you to track milestones by environment.
7. Transform the Excel file into a Status Report by adding columns that will help you track your source control changes such as "Project Collection", "Application", "File Name", "Line Number", "Search Results", "Needs Updated", "Needs Deployed", and "Done".
   1. With a little Excel function magic, you can extract the values for these columns and then hide the Search Results column.
   2. The best part as a table the filters were added for you, so you can filter down. This allows you to delegate the research amongst team members to the Applications they are most familiar with.
8. Research your source code and determine what "Needs Updated"
   1. "Needs Updated": No; "Needs Deployed": No
      1. Poor server name is found in phrases. I hope your new Server Alias you're replacing the server name is more unique. If not, make your case now or you'll be crying next time.
   2. "Needs Updated": Yes; "Needs Deployed": No
      1. It's a comment that needs updated as a reference, but should not be deployed to avoid increasing the scope of what needs tested. Be careful with this scenario, because you really don't want to have inconsistencies between the code you have in source and in production.
      2. You may want to run any scenarios like this with your Solution Architect or other migration team members first. However, this situation is usually acceptable.
   3. "Needs Updated": Yes; "Needs Deployed": Yes
      1. Actual server name change.
9. Create a branch for each Application > check-in your changes
10. When you are asked to deploy > merge your changes to the trunk > request a code review and provide your code reviewer the Excel sheet reflecting what needs updated versus what needs checked in > with approval check in the changes to trunk > and proceed with your deployment process.

# Deployment Process

## InstallShield, Batch, PowerShell, etc.

1. Your deployment process should be checked into source control.
   1. This ensures that you'll find the references when you search through Source Control
2. InstallShield
   1. This represents the desired state of the computer after installation so search for everything as if this was another environment. Again, having this end Source Control greatly simplifies this process.
   2. Environment confusion (Component Services environment example)
      1. The server name may appear in two places
         1. Server Configuration > Component Services > Computers > My Computer > COM+ Applications
            1. This represents what is on the computer, not the desired state.
            2. You shouldn't use this, because your InstallShield shouldn't change environment specific variables. Instead use MSI Command-Line Arguments to pass in the server name.
      2. Media > Releases > Releases > Default Configuration > {Environment} > In the Properties pass in the server name.

## Build/Release Definitions

Take advantage of the History tab to get the entire source of the Build or Release Definition. The Diff shows the changes between the current and previous versions. You don't care about that as much as it shows the entire source of the definition.

Once you identify where the server name is referenced, you should be able to deduce with a little effort where it is edited in the definition.

1. Edit the Build Definition > History > Select first entry > Click Diff
2. Edit the Release Definition > History > Select first entry > Click Diff
   1. This shows all of the web.config token transformations by environment all in one place.

I do not know of a quick way to search through the source of all the Build/Release Definitions. They are stored in the database, so it is possible. So far the amount of time for opening the History tab vs. trying to locate the definitions in the database and how to properly query all of them at once did not seem to be worth it. However, as I see the number of release definitions increasing the prospect may be of interest soon.

## Continuous Integration (Build Definitions)

Continuous Integration is the process of checking in your code and having it automatically build and run unit tests on a Build server letting the team know immediately if code was not fully checked in or checked in with tests failing. Continuous Integration DOES NOT mean you are running integration or regression tests.

Be mindful of integration tests posing as unit tests. Integration tests validate your code against an environment. For developers, that's usually the LOCAL or DEV database, folder/file structure. Unit tests validate your code. While it's fine to have some integration tests, too many and you're duplicating efforts with QA.

1. After checking in your code changes into Source Control your continuous integration tests should pass.
   1. Don't worry at this stage that the old server is still up and running. Actual integration tests will be performed later.

## Continuous Deployment (Release Definitions)

Continuous Deployment is the process of deploying code triggered by actions such as a queued build completing successfully. This process could be very involved including triggers off a successful continuous integration build followed by approval requests sent at the completion of each stage, and tests passing.

1. Pay close attention to the triggers associated with your Release Definitions. You'll want the process to be fairly manual for any migration deployments, so after you queue up a release change all the triggers for that instance to manual.

# Database

## Search for references to server name

Stored Procedures may read files for bulk inserts or save data off to folders after processing. The server name may be stored in a reference table for use by queries or code.

1. Search for references to the server name in:
   1. Tables
   2. Functions
   3. Stored Procedures

## Database Migration

If you are tasked with the migration, here are the basics with more considerations in the next section. This is not a complete list, but gives you an idea of where to start.

1. Work with Operations to install SQL Server on the new server
2. Export/Import databases
3. Setup Linked Servers
4. Setup Permissions

## Database is stored on this server

Trust your Database Administrator (DBA) to perform the migration. However, when issues come up here are a few settings to check from your side. Types and Messages may not be obvious.

1. Identify
   1. Users and Roles
   2. User Defined Types (SELECT \* FROM sys.types)
   3. Custom Messages (SELECT \* FROM sys.messages)
   4. Permissions
   5. Grants on stored procedures and types
   6. Linked Servers
      1. Those servers may need adjusted to allow communications.
      2. These can be very problematic.

## Database Connection Strings

1. As you progress through the rest of the checklist keep a special eye on database connections strings in:
   1. Source Control (your code)
      1. Web.config, Web.Debug.config, Web.QA.config, Web.UAT.config, Web.PROD.config, Web.Tokenized.config, etc.
   2. Reports (SSRS Data Source)
2. Work with the DBA to ensure the user associated with the database connection string has the correct permissions.

# Reports

Trust your Reporting Services Administrator to perform the migration. However, when issues come up here are a few settings to check from your side. These are from the perspective of SQL Server Reporting Services (SSRS).

1. Data Sources should be isolated for reuse by Reports and easier maintenance.
2. Verify that the reports ran correctly on the old server.

# Applications

## Add/Remove Programs

1. Identify installed company applications.
2. Identify installed third party applications.
3. Identify which third applications are required for the company applications to work.
4. Identify which company and third party applications are no longer used on this server.
   1. This includes applications that have been migrated to a different server.

## Programs Files

Applications may be installed, but don't appear in the Add/Remove Programs. Check both "Program Files" (32-bit) and "Program Files (x86)" (64-bit).

1. Identify installed company applications.
2. Identify installed third party applications.
3. Identify which third applications are required for the company applications to work.
4. Identify which company and third party applications are no longer used on this server.
   1. This includes applications that have been migrated to a different server.

## Miscellaneous Folders/Files

1. Identify any applications outside of the Program Files folders.

## Registry

Sometimes registry settings are changed manually or via a "one-time" update. Verify these changes come over.

1. Given your list of company applications, identify where the registry entries are stored and the current registry settings.

# Folders

## Miscellaneous Folders/Files

1. This means you need to tediously review within reason the system's folder structure to find any non-standard usage such as:
   1. Applications may read or write to various folders. Based on timestamps you may get tipped off if it is in use.
   2. Archives may need backed up if not migrated.
   3. Test projects and files or any other "leftovers" typically may be skipped.

# Dependencies

When you use third party applications, frameworks, or anything else you need to learn what they are dependent on. How about tea and crumpets with the Cheshire cat? Be careful Alice, we seem to be going down the rabbit hole in the obscure here.

1. Identify if they require:
   1. Specific DLLs in a specific folder.
      1. Perhaps manually setup.
   2. Specific Services running.
      1. Such as CSLA framework has a dependency on MSDTC (Microsoft Distributed Transaction Manager)
         1. Yeah, you'll uncover this issue the hard way when a Windows Update decides to disable the service for "security" reasons.
         2. Your StackTrace will have a DataPortal error referencing MSDTC.
   3. With some of the old stuff on this server, who knows until you run across it, sigh.

# Security

## Anti-virus

Trust Operations to migrate the Anti-virus. For the obscure connection issues identify any rules to allow the applications to work.

## Firewalls

Trust Operations to migrate the Firewalls. For the obscure connection issues identify the ports that are expected to be open for the applications to work.

1. Inbound Rules
   1. Name, Profile, Program, Protocol and Local Port
2. Outbound Rules
   1. Name, Profile, Program, Protocol and Local Port

## Whitelists and Blacklists

If the server is public facing, Operations may need to work with your Internet provider in identifying whitelists and blacklists. Unless the new server is assigned the same IP addresses internally and externally, IP addresses may change. These IP addresses may be on either whitelists or blacklists inappropriately for the new server, so Operations should review the list if you have connection issues. Also, verify that your external IP addresses are not blacklisted in the anti-spam databases.

1. Trust Operations to migrate the Whitelists and Blacklists.
   1. They will check hardware such as routers and modems.
   2. They will check software such as Barracuda Firewall and VPNs.
2. Verify new external IP addresses are not in the anti-spam databases.
   1. <http://whatismyipaddress.com/blacklist-check>

## Email

Example: Email feature was moved from the website on the abc.com domain to a middleware service on the xyz.com domain already in place to support the primary website on the xyz.com domain.

1. Issue: Emails were failing to deliver to users as the email providers classified them as SPAM. The user could whitelist the email address all they wanted, but they would still not get the email. The email provider was filtering out emails by checking for blacklisted IP addresses in the anti-spam database, "spoofed" domains, etc.
2. Root Cause: The emails from domain abc.com were sent through a middle-ware service under a different domain xyz.com. This type of "spoofing" where the From Domain is different than the domain that sent the email is highly suspicious and is assumed to be SPAM.
3. Solution 1: Send emails from the actual domain. Email From abc.com, sent from abc.com.
4. Solution 2: Send emails from the middleware domain. Email from xyz.com, sent from xyz.com.
5. Solution 3: Revers DNS lookup.
   1. <https://www.whatismyip.com/reverse-dns-lookup/>
6. Afterward: Your IP address may be blacklisted for these actions. You can wait for the IP address to naturally be dropped over time or proactively try to be "unblacklisted".
   1. <http://whatismyipaddress.com/blacklist-removal>

# Cleanup

As part of the migration, you may need to deploy new code. Use an application like [Beyond Compare](https://www.scootersoftware.com/) to compare old and new server application folders to verify that there are no residual files.

Example: A website was migrated. However, Business wanted the site deployed with recent CSS changes to give Value Add to the clients for dealing with the potential outage during the server migration. This is usually covered in the deployment tickets and was not included with the migration. Beyond Compare to the rescue.

1. Issue: QA discovered that the website's styles were incorrect.
2. Root Cause: The developers deleted the old CSS stylesheets on the DEV servers, so the new CSS files could be used. The application's deployment tickets were accurate and great. However, the migration team was not aware of the steps due to miscommunications.
3. Solution: Use Beyond Compare or another folder/file comparison tool performing binary compares to ignore timestamps. With this information the developers immediately realized the steps were missing and the migration steps were updated.

# Rollback Plan

Everything works exactly the way you want the first time? You may want to wake up from your pipe dream and join the real world. Regardless of all the painstaking hours you have devoted to this migration project, something will go wrong. It always does. I judge character not by how well someone got it right, but how well they recovered when a mistake was made. Be prepared.

If you are not migrating to a new machine of virtual machine (VM), then you'll want to create backups and snapshots. There will be many changes and you need to roll back to where it was beforehand.

As a team decide how you will rollback when the inevitable happens.

1. Multiple servers
   1. Rollback everything
      1. If multiple servers have to be migrated at the same time you may have to rollback all of the servers for an environment.
         1. Example: The Database server failed to migrate. The new Reporting Services server isn't compatible with the old server or there are too many server name changes to warrant a partial rollback.
   2. Rollback an application or database server
      1. If you are using an alias for the server and there are no compatibility issues, then you may change just the alias back to the old server. You would definitely want to retest in the previous environments. While Chuck Norris may be able to get away with testing in production, you're not Chuck Norris.
2. Physical machine
   1. Migrating to new machine
      1. Turn the new server off.
      2. Turn the old one on.
   2. Else (ex. Domain migration)
      1. Make a backup of the server. Use a tool like Acronis to clone the drive to another hard drive or make a backup.
      2. Restore the backup at rollback.
3. Virtual machine (VM)
   1. Migrating to new VM
      1. Turn the new VM off.
      2. Turn the old VM on.
   2. Else (ex. Domain migration)
      1. Take a snapshot of the virtual machine before deployments.
      2. Restore the virtual machine from the snapshot.
4. Update server alias.
5. If it's a domain migration, you'll migrate the alias/server name back to the old domain.

# Developer Testing

## Communication

1. Keep the migration team in the loop. When it comes time for deployments, communicate to anyone who uses the server when you will be performing the deployment.
2. QA should respond to the deployment emails letting everyone know if the tests passed or not.
3. If you need to rollback, respond to the deployment emails.
4. I recommend that you create a distribution group for the migration team and another for deployments.

## Identify how every effected application and their dependencies are used

1. With all of the information gathered regarding what uses and is used by the server, you need to review with QA to determine what is already covered by the regression tests.
2. Anything not covered by the regression tests or aren't confident it's the same feature should be researched in Source Control just like you did above.
3. Work with QA to figure out how to properly test or confirm that it is a feature that is no longer supported. As with everything document these decisions.

## Turn off the old server

May seem obvious, but all too often developers must run their own tests against the new server while the old server is still running. Once everything is working as expected, ask Operations to turn off the old server. Run your simple smoke tests before turning over the testing to QA. After all, if you immediately find that your smoke tests fail when the old server is turned off, you will save QA from wasting efforts.

1. Turn off the old server.
2. Run developer smoke tests and resolve any issues found.
3. Notify the distribution list that QA may start testing.

# QA Testing

## Create test plans

As a developer you are to assist the QA staff by answering any questions they may have and relay to them the tests you have already performed. This will give them an idea of what else to test, so they can create any additional tests not covered by their regression tests.

1. QA will create test plans that should test anything affected by the server changes.
2. Review the test plans to ensure:
   1. Nothing was missed.
   2. They are testing enough.
   3. They are not testing too much.

## Update manual and automated regressions tests

As QA starts to develop automated regression tests, you may be asked to assist. A common way is with Gerkin and SpecFlow which follows the process of defining steps that need to be performed from the user's perspective in order to navigate to and then test. <https://specflow.org/documentation/Using-Gherkin-Language-in-SpecFlow/>

1. Following DevOps principles, integrate into the release pipeline the stable automated regression tests. This can range from running batch files, powershell, to your Build/Release definitions.
   1. The Build definition would need to build the SpecFlow tests environment agnostic and include them into the Build Artifacts.
   2. The Release definition should not deploy the SpecFlow artifacts.
   3. The Release definition would need to run the SpecFlow tests after the application artifacts have been deployed to an environment.

## DEV/SIT/UAT/PROD

Depending on how well your environments are structured such as similar hardware, or how long automated regression takes, you may need to change what is tested by environment.

## DEV

Developers should already have unit tests and integration tests in place, however, QA may want a basic smoke test that performs the most core functionality. An example for an eCommerce site would be to create an order, process payment, and send an invoice through email.

## SIT

Quality Assurance (QA) team perform there tests in the System Integration Test (SIT) environment. QA should hammer this environment with all of their automated and manual tests.

## UAT

This is a subset of actual users test in the User Acceptance Test (UAT) environment. The Business team should identify the testers. QA may provide the users a copy of the QA test plan so the users know what to test and what has not yet been tested to create their own test plan. This environment is the final gate before deployment to production.

## PROD

This is the production server or client machines. Production servers are typically flipped over on the weekends to avoid outages during the business week. Light testing is done in the production environment. If testing will cause issues such as adding invalid orders, make sure you can cancel the orders or have a planned database refresh and cleanup of any artifacts.

## External (public facing)

If the server is public facing, meaning that users may access the server or web site outside of the company's network, then you need to test this environment as well. Preferably, if PROD is available externally, then the other environments should be as well. Your company may want to restrict the IP addresses that may connect to the server by blacklisting everything except IP addresses that developers and QA staff are using.

1. Setup a computer outside the company network.
   1. Login from home, hotel, library, etc.
   2. Or connect through a hotspot or a separate dedicated external modem/router.
   3. Or create a virtual machine (VM) in Azure or other provider that is external.
2. Verify the computer is outside the network
   1. In a command prompt (cmd) > ipconfig > Typically, the IP is the IPv4 Address
   2. Open <https://whatismyipaddress.com/> and enter your IP address
3. Run the same tests you did for the internal environment. Everything should return the same results.
   1. The goal is ensure that connections will still work for our clients.

# Conclusion

Server migrations can be very involved, frustrating projects, but are a necessary evil to stay current with technology. Whenever possible, consider taking advantage scalable products such as Azure. Regardless of migration to the cloud, VM, or another physical computer, this checklist with notes and examples should help you identify most of what you'll need to perform the server migration.