Chart, radar chart

Description automatically generatedChart, radar chart

Description automatically generatedChart, radar chart

Description automatically generated



Shape

Description automatically generated with medium confidence

|  |  |
| --- | --- |
| **EfficientIA Solutions** | **Software Change Management**  Where is my latest Backup |
|  |  |

|  |  |  |
| --- | --- | --- |
| Document No.: |  |  |
| Revision No.: |  | Rev 0 |
| Author: |  | Carel de Jager / Jai Porter-Scott |
| Effective Date: |  | 30/07/2022 |
| File Name: |  | EFF - Ignition SCM Rev0 20220730 |
|  |  |  |
| Project Number: |  |  |
| Project Description: |  | Software Change Management |
| Client: |  | EfficientIA Solutions |
|  |  | Unit 8, 707 Forest Road  Peakhurst 2210  Contact: Carel de Jager  Phone: +61 437411477  [carel.dejager@efficientia.com.au](mailto:carel.dejager@efficientia.com.au) |



|  |  |  |
| --- | --- | --- |
| Distribution List | | |
| Name | Company | Responsibility |
| Gordon Cornish | EfficientIA Solutions | Managing Director |
| Richard Anderson | EfficientIA Solutions | Business Development Manager |
| Carel de Jager | EfficientIA Solutions | Advances Solutions Manager |
| Jai Porter-Scott | EfficientIA Solutions | Technical Services Engineer |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Revision | | | Rev 0 | | | | | | | | | | | |
| Author | | | Carel de Jager / Jai Porter-Scott | | | | | | | | | | | |
| Reviewed By | | |  | | | | | | | | | | | |
| Approved By | | |  | | | | | | | | | | | |
| Total Pages | | |  | | | | | | | | | | | |
|  | |  | | |  | | |  | |  | | |  |
| History | | | | | | | | | | | | | |
| Rev | Comments | | | | | Author | | Reviewed | Approved | | Date | | |
| 0 | Initial Release | | | | | RA | | CDJ |  | |  | | |
|  |  | | | | |  | |  |  | |  | | |
|  |  | | | | |  | |  |  | |  | | |
|  |  | | | | |  | |  |  | |  | | |
|  |  | | | | |  | |  |  | |  | | |
|  |  | | | | |  | |  |  | |  | | |
|  |  | | | | |  | |  |  | |  | | |
|  | |  | | |  | | |  | |  | | |  |
| Client Approval | | | | | | | | | | | | | |
| Name | | | | Project Responsibility | | | Signature | | | | | Date | | |
|  | | | |  | | |  | | | | |  | | |
|  | |  | | |  | | |  | |  | | |  | |

Notes for Rev 0 :

This document is for the use of Flow Systems and is not to be distributed to any 3rd parties without written consent from EfficientIA.

This document is for the use of Flow Systems and is not to be distributed to any 3rd parties without written consent from EfficientIA.

This document is for the use of Flow Systems and is not to be distributed to any 3rd parties without written consent from EfficientIA.

This document is for the use of EfficientIA Solutions and is not to be distributed to any 3rd parties without written consent from EfficientIA Solutions.

TABLE OF CONTENTS

[1 Introduction 4](#_Toc111636602)

[2 Abbreviations 4](#_Toc111636603)

[3 What is Software Change Management? (MDT Autosave) 5](#_Toc111636604)

[4 Tools and vendors 6](#_Toc111636605)

[4.2 Git (git, n.d.) 6](#_Toc111636606)

[4.3 PLC Specific Software Change Management (MDT Autosave) 8](#_Toc111636607)

[5 Ignition Structures (Inductive Automation, 2021) 9](#_Toc111636608)

[5.1 Gateway Configuration 9](#_Toc111636609)

[5.2 Tags 9](#_Toc111636610)

[5.3 Projects 10](#_Toc111636611)

[6 GitHub Structures and Interface 11](#_Toc111636612)

[7 Proposed Deployment Environments & Architectures (Inductive Automation, 2021) 12](#_Toc111636613)

[7.2 Development 12](#_Toc111636614)

[7.3 Testing 14](#_Toc111636615)

[7.4 Production 14](#_Toc111636616)

[8 Software Change Management Procedures 15](#_Toc111636617)

[8.2 Step 1 - Using the Enterprise Administration Module 15](#_Toc111636618)

[8.2.1 Collect and Restore Backups Automatically (Inductive Automation, 2022) 16](#_Toc111636619)

[8.2.2 Move Changes and Projects between different Environments 20](#_Toc111636620)

[8.3 Step 2 – Enterprise Administration Module with Git tools 25](#_Toc111636621)

[8.3.1 Start with an existing Ignition project 26](#_Toc111636622)

[8.3.2 Send my project resources to the Online GitHub Repository 27](#_Toc111636623)

[*8.3.3* GitHub Online (*https://github.com/)* 29](#_Toc111636624)

[8.3.4 Get a project backup from the GitHub Online 30](#_Toc111636625)

[8.4 Step 3 – Fully Integrated Git Technology 32](#_Toc111636626)

[9 References 33](#_Toc111636627)

[10 Appendix 34](#_Toc111636628)

[10.1 Install GitHub desktop and register online 34](#_Toc111636629)

[10.2 Release Project Versions (Preston-Werner, n.d.) 35](#_Toc111636630)

# Introduction

This guide is aimed to help you better understand how to deal with deploying Ignition applications and provides some best practices for setting up your development and testing workflow. Having a solid deployment workflow allows for better change management, minimizes mistakes, and leads to a more productive team.

This guide helps Ignition users who are looking to set up a scalable, flexible, enterprise-grade system. Most users start from a different architecture or different model and move toward proper SCM (Software Change Management) where other users may never choose to employ all recommendations.A picture containing text

Description automatically generated (Inductive Automation, 2021)

The devices to manage are typical PLC (programmable logic controllers) and other industrial equipment propriety protocols and file systems.

With the IT/OT convergence in the last decade, more tools are available which forces companies to change to more IT standards like Git.

Although Git tools do not understand proprietary file systems and devices, the market moves towards these tools due to pricing pressure.

Git technology fits very well with Ignition which uses standard and open file system.

These IT technologies do not come naturally to the traditional OT (Operational Technology) personnel and hopefully this document will help them to follow and understand the procedures and functionally. Thus, this document will **concentrate on the GIT technology to manage Ignition files and projects.**

This document is a **guideline**, and it is essential that you **create your own procedures and regulations** which are **easy to follow** and subsequently sustainable.

# Abbreviations

|  |  |
| --- | --- |
|  |  |
| SCM | **S**oftware **C**hange **M**anagement |
| CMS | **C**hange **M**anagement **S**ystems |
| PLC | A **P**rogrammable **L**ogical **C**ontroller is an industrial computer that has been ruggedized and adapted for the control of manufacturing processes, such as assembly lines, machines, robotic devices, or any activity that requires high reliability, ease of programming, and process fault diagnosis. (Wikipedia, n.d.) |
| HMI | **H**uman **M**achine **I**nterface is a –computer interaction, a user interface (UI) is the space where interactions between humans and machines occur. The goal of this interaction is to allow effective operation and control of the machine from the human end, while the machine simultaneously feeds back information that aids the operators' decision-making process. |
| CNC | **C**omputer **N**umerical **C**ontrol machining definition is that it is a subtractive manufacturing process that typically employs computerized controls and machine tools to remove layers of material from a stock piece. (Thomas, n.d.) |
| IT |  |
| OT |  |
| HTML |  |
| JASON |  |
| EAM |  |
| SQL |  |
| OPC |  |
| MQTT |  |
|  |  |

# What is Software Change Management? (MDT Autosave)

Georgetown University defines change management as the complete **set of processes** employed to ensure that changes are implemented in a visible, controlled, and orderly fashion. Focus should be placed on several key elements:

1. **Process:** Must be a structured method that stays the same
2. **Completeness**: Must include all objects (person, place, or thing)
3. **Visibility:** Without visibility, Change Management does not exist
4. **Control:** Who, what and where
5. **Order:** Changes may need a particular sequence to be effective Change Management as it applies to plant automation focuses on the control systems that operate the production equipment. Change Management System (CMS) applications have matured over the past 20 years along with the increases in sophistication and capability of the automation devices and control software developed by automation vendors. There are many elements of a CMS that make its function distinct from single-file-based source code control systems, with the most distinct differences being a suite of tools to manage a group of files (often referred to as a “project”) as a single entity, and processes for detecting source code changes outside of source code control. In most automation devices the entire project file is necessary when managing revisions and identifying changes.

**Key Change Management Functions in the OT environment**

Diagram

Description automatically generated

Figure 1 - Typical Flow (Auvesey)

* Version control
* Program backup
* Change detection
* Historical record of changes
* Common central repository
* Secured user access
* Program comparison

Deployments should be treated as part of a development workflow, not as an afterthought. Your workflow will usually include at least three environments: Development, Testing, and Production. In that case, the workflow might look like this:

* 1. Developers work on bugs and features in a development environment.
  2. Once features are implemented, they are merged into the testing branch and deployed to the testing environment for quality assurance and testing.
  3. After testing is complete, the development branch is merged into production and then deployed to the Production environment. (Inductive Automation, 2021)

Icon

Description automatically generated

Figure 2 – Flow (Inductive Automation, 2021)

# Tools and vendors

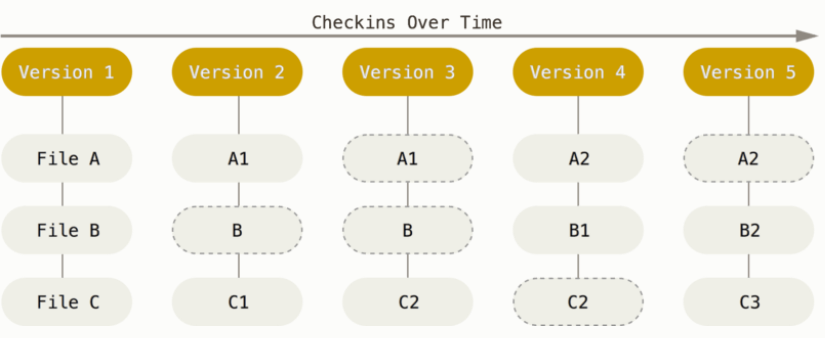
There are a wide range of tools and solutions in the market but basically two different vendors to mention in this document. The one is GIT technology and the second one is PLC specific vendor software. Git concentrate more on general file systems where PLC specific solutions understand the propriety files and protocols for one or many different brands. The most popular ones are **Versiondog** from Auvesy (now called Octoplant) and **MDT Autosave**. (Recently acquired from Auvesy)

**We will concentrate on Git which lends itself well with the Ignition open file systems.**

## Git (git, n.d.)

Git stores and thinks about information in a vastly unique way and understanding these differences will help you avoid becoming confused while using it. Conceptually, most other systems store information as a list of file-based changes.

Git thinks of its data more like a series of snapshots of a miniature filesystem. With Git, every time you commit, or save the state of your project, Git basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot. To be efficient, if files have not changed, Git does not store the file again, just a link to the previous identical file it has already stored. Git thinks about its data more like a stream of snapshots.



Most operations in Git need only local files and resources to operate — generally no information is needed from another computer on your network. Because you have the entire history of the project right there on your local disk, most operations seem almost instantaneous.

Everything in Git is **checksummed** before it is stored and is then referred to by that checksum. This means it is impossible to change the contents of any file or directory without Git knowing about it. This functionality is built into Git at the lowest levels and is integral to its philosophy. You cannot lose information in transit or get file corruption without Git being able to detect it.

Git has three main states that your files can reside in: **modified**, **staged**, and **committed**.

* **Modified** means that you have changed the file but have not committed it to your database yet.
* **Staged** means that you have marked a modified file in its current version to go into your next commit snapshot.
* **Committed** means that the data is safely stored in your local database.

This leads to the three main sections of a Git project: the **working tree**, the **staging area**, and the **Git directory**.

Diagram

Description automatically generated

The **working tree** is a single checkout of one version of the project. These files are pulled out of the compressed database in the Git directory and placed on disk for you to **use or modify**.

The **staging area is a file**, generally contained in your Git directory, that stores information about what will go into your next commit. Its technical name in Git parlance is the “index,” but the phrase “staging area” works just as well.

The **Git directory** is where Git stores the metadata and object database for your project. This is the most important part of Git, and it is what is copied when you clone a repository from another computer.

This is the basic Git workflow:

* 1. You **modify** files in your working tree.
  2. You selectively **stage** just those changes you want to be part of your next commit, which adds only those changes to the staging area.
  3. You do a **commit**, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

If a particular version of a file is in the Git directory, it is considered **committed**. If it has been modified and was added to the staging area, it is **staged**. And if it was changed since it was checked out but has not been staged, it is **modified**.

You can find more detail at <https://www.git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository#ch02-git-basics-chapter>

## PLC Specific Software Change Management (MDT Autosave)

An automation Change Management System is a centralized system that **manages changes** to **program logic** for controls programs and devices such as PLCs (programmable logic controllers), CNCs, HMIs (Human Machine Interface), PC control systems, robots, drives and general automation programs. A typical small plant will have a few hundred programs that should be managed, while large plants will have several thousand. Over the life of a facility the investment in program logic alone represents a significant expenditure that should be preserved and optimized. In order to do this a CMS should have the following features:

* Diagram, schematic

  Description automatically generatedAn archive of prior revisions of programs
* The ability to detect changes
* Tools for documenting changes and making them visible to users
* A historical record of who made the change, when, and from where it was made
* Secured user and workstation access
* Features for controlling editor operations mapped to user permissions
* Procedures for recovering from hardware failures
* Change notification

# Ignition Structures (Inductive Automation, 2021)

Ignition is **server-based** software and **installed on a central server**. That means all of Ignition’s configuration is stored

on the server. It is one place to install, license, configure, backup, and manage. Client applications are downloaded

from the Ignition server (Vision) or are native HTML (Perspective) and as a result are automatically updated as new

changes are deployed.

When you are developing in Ignition, there are 4 main areas of configuration:

* Gateway Configuration
* Tags
* Images
* Projects

Each of these areas are **stored differently on the server**. The **Gateway backup contains the configuration for all of these**

but **some can also be handled individually**. It is critical to understand the differences and how we interface with them

when deploying changes.

## Gateway Configuration

The **Gateway configuration** area includes all the different settings, profiles, and connections you edit in the

**Configuration section** of the Gateway webpage. This is where most of the settings that affect the whole Gateway are

set up. We can add database and device connections, users, and roles, adjust alarm settings, set up security, create a

schedule for a Gateway backup to be taken automatically at specific times, and much more.

All these settings are **stored inside of Ignition’s internal** **SQLite database**. They are only accessible by the Gateway

user interface. Outside of a full Gateway backup, you cannot export or import these settings from Gateway to Gateway.

Ignition’s EAM Module currently does not support moving these settings from Gateway to Gateway either. This makes

it tricky to work with when developing our deployment strategy since all changes must be made manually. It is also

difficult to track changes since everything is inside of an SQLite database.

Gateway backups are not normally used when migrating changes because they are all-or-nothing. Different environments often have separate Gateway configuration settings for items such as devices, databases, and authentication. We recommend creating a process that all developers follow to track changes of these settings outside of Ignition and to document the differences in these settings within each environment (development, testing, and production).

## Tags

Graphical user interface, application, table, Excel

Description automatically generated

Tags are stored inside of **Tag Providers**. Tags are **not part** of an Ignition Project. Rather, projects simply reference tags.

A Tag Provider is a collection of tags (a tag database) and can be local or remote. An Ignition server can have one or

more tag providers associated with it. Tag Providers can provide logical groupings of tags (separate stores of tags). By

default, Ignition is shipped with a local tag provider called “default.” **Local tag providers are stored inside of Ignition’s internal SQLite database**.

Ignition can **export and import** Tag configurations to and from the **JSON** (JavaScript Object Notation) file format. You

can import XML (Extensible Markup Language) or CSV (Comma Separated Value) file formats as well, but Ignition

will convert them to JSON format.

Tags can also be moved through Ignition’s EAM Module from Gateway to Gateway. Tags are much easier to work

with when deploying changes. However, tags are still stored inside of Ignition’s internal SQLite database and

steps must be taken to track changes of tags. We recommend continuously exporting tags to a JSON file and committing the file to a source control repository to track changes.

## Projects

Projects are the main unit of configuration in Ignition. Projects hold all the designed elements that do the real work.

Your projects can hold both interactive elements (like controls, charts, reports, entry forms, and more) and persistent

elements (like historical loggers, automated reports, etc.).

In Ignition, a project is a unit of configuration that contains:

* Vision Windows & Templates and Perspective Views: Screens users interact with
* Transaction Groups: A bi-directional link between databases and PLCs
* Reports: PDF reports for displaying and recording data
* Scripts: Timer and event-based scripts used throughout the system
* Alarm Notification Pipelines: Pipelines that notify users of alarm conditions
* General settings and properties: The settings that control access, resource connections, layout, timing, and more.
* And much more

You use the Ignition Designer to configure and create projects. The projects are then viewed in the runtime (Vision

Clients or Perspective Sessions). You can create as many projects as you want, and users can easily jump between

projects on the fly or open multiple projects at the same time.

Projects are stored in the file system as a series of folders and files. Some files are binary, while others are stored as

plain text files. Projects are stored inside the data directory on the Ignition server.

**C:\Program Files\Inductive Automation\Ignition\data\projects**

Since projects are stored in the file system, it is possible to use best-in-class source control tools outside of Ignition

to track changes of projects, such as Git. Changes in the Ignition Designer get updated in the file system. Ignition also

continuously monitors the file system for changes and automatically reads in the changes.

Projects can also be imported and exported through a file (.zip extension) and moved from Gateway to Gateway

using EAM.

We recommend using an external source control tool to track project changes, such as Git. **You can turn the**

**project folder into a repository** and use tools to deploy changes from development to testing to production

environments outside of Ignition.

**To Summarise**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Gateway Configuration | Tags | Images | Projects |
| Change Tracking | Manual Documentation and  Gateway Backup, then commit to Version Control | Tag Export, then commit to Version Control | Image Export, then commit to Version Control | No export required. Auto-commit project folder changes to Version Control |
| Backups | Gateway Backup | Gateway Backup | Gateway Backup | Gateway Backup |

# GitHub Structures and Interface

# Proposed Deployment Environments & Architectures (Inductive Automation, 2021)

Now that we have a good idea of how Ignition’s and Git configurations is stored and works, let us take a closer look at each

environment to see what the most efficient ways are to deploy to each one of them. In the reference architecture

below you will see all 3 separate Ignition environments (development, testing, and production) along with Ignition’s

EAM Module and a source control system such as GitLab.

Diagram

Description automatically generated

Figure 3 - Ideal Architecture and Procedures

## Development

The development environment (dev) is the environment in which active Ignition development is occurring. This environment is the first step of the workflow. Here developers are working on bug fixes and new features before introducing them to the production environment. It is very handy to have a separate branch in your source control

system called development to represent your development environment.

There are 2 main camps on how to approach the development environment: **using a shared development environment**

or using **individual workstations** as their own development environment. A shared development environment is where you have a central Ignition server dedicated to development that all developers work on.

An individual development environment is one where each developer has Ignition installed on their workstation with

their branch of the project. There are pros and cons to each approach. **Ignition supports concurrent development**

and most customers tend to go with a **shared development** environment, so **nothing must be installed on each**

**developer’s workstation**. They can just launch the Designer from the development server. If an individual development

environment approach is taken, each developer’s workstation will need to have its Ignition Gateway configured

separately. Regardless of the approach, changes will need to be committed to the source control repository and

deployed through the workflow.

Consideration should be taken on the commercial cost of licensing to maintain this architecture. Ignition is downloadable and fully functional without any license installed. It will run for two (2) hours after you need to reset the system. It does not affect the development environment.

Off course, you need a license if you need to test functionality for more than two hours. Keep in mind that discount options are available for test systems.

**Configuration considerations between different environments**

It is extremely important to understand the Ignition differences in each environment. **They cannot be the same**. The production environment is communicating to live data sources such as devices and SQL databases and more. You may **not want your development and testing environments connected to the same live data sources**. You also may have **different authentication sources** or permission models on the development and testing environments. Here is a small list of common differences:

* Devices or PLC connections: Using Ignition’s native drivers
* OPC UA connections: Connecting to third-party such as Kepware or a PLC directly
* Database connections: Connecting to a SQL database such as MySQL or MS SQL
* Authentication profiles: Different authentication sources or especially a different admin or root password. Sometimes you want to lock out access to the Designer and Gateway in the production environment.
* Alarm notification profiles: Connections to email, SMS, or voice.
* Redundancy settings: Often you have redundancy on production and not development or testing.

We recommend keeping the **same connections (same name) on each environment but with different IP addresses** or settings behind the scenes. That way any resources dependent on knowing the name will still function from environment to environment. For example, you may have a dedicated SQL database for each environment. All 3 Ignition Gateways will have the same database connection by name, but they will point to their dedicated SQL database that may be local or remote. Screens or named queries using that connection will function across all environments since they are referencing the connection by name.

Document these differences to ensure you do not deploy a change to production that points to a development or

testing system.

**How do I simulate real device data on development and testing?**

This is a very common question since you want to have access to live data in all environments. Sometimes there are limitations or performance issues that must be understood. For example, some PLCs do not support more than 1 connection or would get saturated with multiple Ignition servers polling the same tags multiple times. There are several ways to accomplish getting access to real data:

* **OPC UA:** You can connect your development and testing environments to your production environment through OPC UA. The development environment would have the same OPC server name, but it would point to production instead of pointing to itself. Be careful, as it is possible for development and testing to affect production, especially with writing to tags.
* **MQTT (Message Queuing Telemetry Transport):** This is the recommended way of getting live data to all environments. You can have your production environment publish all tag data to a central MQTT server using Ignition’s MQTT Transmission Module. All 3 environments can subscribe using Ignition’s MQTT Engine Module and tags are automatically discovered. Each environment can subscribe without affecting the other. Again, be careful about potentially writing to live PLCs. MQTT servers have ACLs (access control lists) you can define to make the development and testing environments read-only if needed.
* **Gateway Network**: You can connect your development and testing environments to your production environment through Ignition’s Gateway Network and set up remote tag providers. The development environment would have the same tag provider name, but it would point to production instead of being local. Again, be careful about potentially writing to live PLCs. Ignition’s Gateway Network has service-level security to make the development and testing environments read-only if needed. This method is not generally recommended.
  + - **Device Simulation:** 
      * **Programmable Device Simulator:** This is perhaps the best option if you do not need live PLC data. Instead, we can use Ignition’s programmable device simulator to simulate PLC data. The simulator allows you to define any tag path you want along with static or dynamic values. Your development and testing environments will use the simulator while the production environment will use the real PLC. As long as you keep the name of the device the same, the tags will be identical across all systems. See the documentation for more details on the simulator: <https://docs.inductiveautomation.com/display/DOC80/Programmable+Device+Simulator>
      * **SFCs (Sequential Function Charts):** If you need more advanced device simulation, SFCs offer deep flexibility. The Programmable Device Simulator provides configurable signals, but the signals do not change based on a user’s actions. SFCs, paired with memory tags, provide a great way to simulate basic PLC functionality like HOAs and valve statuses, flows, and more.
      * **PLC Software Simulator**: Many PLC manufacturers offer a soft PLC/emulator that can run a real PLC program. If Ignition can connect to the software via OPC, this can be a good option to have the real PLC program with all the tags running in a system that is completely disconnected from the real equipment
    - **Additional PLCs:** If you have a system with a small number of PLCs, it can sometimes make sense to have separate physical PLCs for the development and QA environments, especially if PLC programming changes may be rolled out at the same time as Ignition project changes.

**What about custom SQL tables that I want synchronized across each environment?**

As we discussed earlier, each environment typically has its own dedicated SQL database. You do not want development and testing working off the same database as production. Generally, this is not a problem for Ignition’s built-in history, alarming, auditing, and transaction groups. However, if you build your own custom tables and interface Ignition with those tables, we must think about how to ensure those tables (structure) and possible data exist in all environments. **The easiest is to create a separate Database BUT make sure you name the connection the same as in the Production environment.**

## Testing

Once the features are implemented and considered stable, they get merged into the testing branch and then

deployed to the testing environment. This is when quality assurance kicks in: testers go to testing servers and verify

that the project works as intended.

It is very handy to have a separate branch in your source control system called testing to represent your testing

environment. It will allow developers to deploy multiple branches to the same server simultaneously, simply by

merging everything that needs to be deployed to the testing branch. It will also help testers understand what exactly is

on testing servers now, just by looking inside the testing branch.

## Production

Once the feature is implemented and tested, it can be deployed to production. The production environment is also

known as live, particularly for servers, as it is the environment that users directly interact with.

**We recommend always deploying major releases to production at a scheduled time, of which the whole team**

**is aware.**

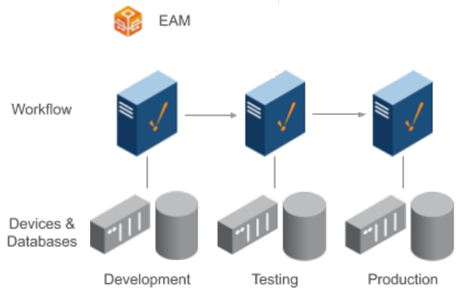
# Software Change Management Procedures

Figure 3 and the discussion above is the ideal scenario with fully SCM procedures in place. That is moving projects from the development environment through to Production AND tracking requests and changes for all procedures.

OT personnel should be aware of SCM, but the Git environment is probably something new and will take **practice and leadership** to sustain proper change management procedures. Therefore, we decided to break the journey up in three parts.

* 1. Using the **Enterprise Administration Module** to move projects between environments. Except for scheduled backups, no change management forms part of the procedures.
  2. Combine the **Enterprise Administration Module with Git** which will add change management to the individual environments.
  3. **Fully integrated Git technology** using the Git tools from the change request through to Production. The possibilities are endless, depending on the OT/IT maturity. We will also start to introduce automated procedures inside Ignition when changes are committed or saved.

## Step 1 - Using the Enterprise Administration Module



The EAM Module provides a secure and intuitive way to manage many Ignition installations from one location. It is ideal

for large enterprises that deploy multiple Gateways across vast geographical distances, but even companies with a

few Ignition Gateways on a single plant floor can benefit from its ability to monitor performance, and automate backup

and recovery from a central location.

With the EAM Module installed, an Ignition Gateway can function as the central Controller Gateway that connects

with, monitors, and manages any number of remote Agent Gateways. Use the Controller Gateway to coordinate and

automate many administrative tasks for Agent Gateways, including:

* Check health and diagnostics
* Assigning, updating, activating, and deactivating licenses
* **Deploying new and updated modules**
* **Disaster recovery**
* **Remote backups**
* Remote Ignition upgrade

You should use all the functions, but we will concentrate on the **transfer, backup**, and **recovery** of projects and changes.

For a complete guide, go to <https://docs.inductiveautomation.com/display/DOC81/Enterprise+Administration>

### Collect and Restore Backups Automatically (Inductive Automation, 2022)

The Backup and Restore tasks are an extremely important part of the Enterprise Administration Module. Performing a Collect Backup is required if you want to do an Agent recovery.

**Note:** For these steps to work you need to configure the network configurations first.

There is **little difference** between an Outgoing and Incoming connection: these terms simply indicate which server the connection was configured on and are mostly ignored by the rest of Ignition.

**But it is good to have a standard connection philosophy** to make fault-finding easier. We recommend configuring on the **higher-level gateway (Controller) as an outgoing connection** to the lower-level gateways. (Edge Gateways)

Follow these steps to establish network communication between the servers. (<https://docs.inductiveautomation.com/display/DOC81/Gateway+Network> )

**Collect Backup**

**Collect Backup** is a Gateway Task that performs a Gateway backup on the selected Agent's machine. Additionally, this task will archive copies of the Agent's modules.

Collect Backup times out by default after waiting 60 minutes for the Agent to generate the backup, plus another 60 minutes for the Agent to transfer the backup to the controller over the Gateway network if the backup generation did not timeout. Transfer timeout is configurable using the system property ignition.eam.task.collectBackup.transferTimeout.

To create a new task, or to view running tasks, navigate to **Config / Enterprise Administration / Agent Tasks**. The Gateway Tasks page will display scheduled tasks, currently executing tasks, unscheduled tasks that can be run on demand, and task history.

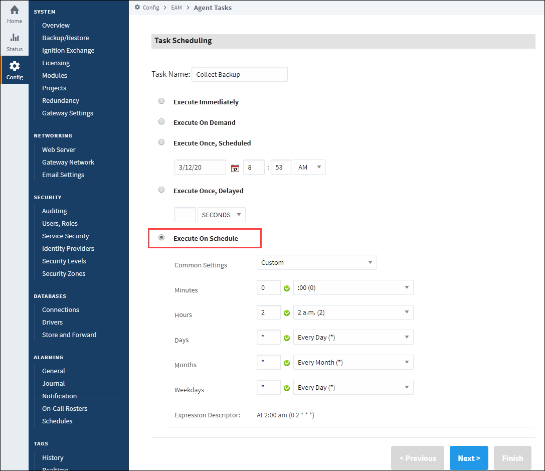
1. To create a new task, click **Create** **new** **Gateway** **Task**. The EAM wizard will guide you through each step of a Gateway task.

Graphical user interface, text, application

Description automatically generated

1. Select **Collect Backup**. After you selected a type of task, click the **Next** button.  
     
   Graphical user interface, text, application, email

   Description automatically generated
2. Set the task name to a descriptive name of your choice.
3. Select **Execute on Schedule** and click **Next**.



1. Next, select the agent groups or individual agents that will be included when the task executes. You can select agents that are not currently connected, but you cannot select pending agents.
2. Click the **Finish** button to schedule the task.  
     
   Graphical user interface, text, application

   Description automatically generated

**Gateway Task Results**

Gateway task execution is reported in the **agent\_events table** in the controller’s configured database. The individual result of the task execution for each agent is report as a separate entry. For example, the following query will return the results of all task executions:

*select \* from agent\_events where event\_category = 'task' order by event\_time desc;*

A task that completed normally for an agent will contain the text NORMAL in the event\_level field. If an error occurred for an agent, the event\_level field is set to ERROR, and the error message is recorded in the message field.

**Running a Task Outside its Schedule**

You can run any task outside the task’s normal schedule by clicking the **More** button and “**run now**” link on the right side of the task. Keep in mind:

* Tasks that have been scheduled to run immediately are already running, and the “run now” link will not be visible.
* A task that has been scheduled to execute only once at a scheduled time or after a delay will be rescheduled to run immediately. The task will not run again after it has been rescheduled.
* A scheduled task will continue to run at its normally scheduled time, even if you run the task now.

**Restore Backup**

**Note:** When restoring from a backup using this task, the newly restored agent will retain its previous name.

1. On the **Controller,**navigate to **Config / Enterprise Administration / Agent Tasks**
2. Click on the **Create new Gateway Task**link.
3. Select the **Restore Backup and** press **Next**.

Graphical user interface, text, application, email

Description automatically generated

1. Schedule the **Restore Backup** task from any of the listed scheduling options. This example uses **Execute Immediately and** press **Next**.
2. Select the **Agent and** press **Next**.
3. If your Archive Path was set to Manual when you created your Controller, you will have to select the **Choose File** button, and navigate to the folder on your computer to locate the Gateway Backup file. If you are unsure about your Archive Path, refer to your Controller Settings. (<https://docs.inductiveautomation.com/display/DOC81/Creating+a+Controller>)

Graphical user interface, text, application, email

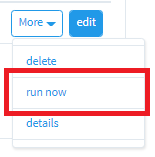
Description automatically generated

1. Once you choose the file, click **Open.** Click **Next** on the Restore Backup task.  
     
    Graphical user interface, text, application, email

   Description automatically generated
2. Review the Restore Backup Task Summary and click **Finish.**Once the Restore Backup task completes, you will receive a successful message from the Gateway task.
3. Now go to your Agent and click **Home**. It takes a few moments to bring up the Gateway. You will see a progress bar while the Gateway is starting up. Once it is complete, you will see all your projects uploaded and ready to go.

**To Summarize**

1. **Schedule a task to** **backup your Project**.
2. **Important**: Also run the Scheduled task **before and after any changes are made**.



1. Remember to **backup your Databases and 3rd party I/O server information separately**.

### Move Changes and Projects between different Environments

The EAM can also **move projects and project resources between agents and the controller**. We will use these

functions to move the resources from the **development** server to the test and **production** server.

We will have a record of when and how the resources moved but **no change management functions** which will provide more detail on the person, task name and what was changed.

**Send Project Task**

On the **Controller**, under the **Config** section of the Gateway Webpage, go to **Enterprise Administration > Agent Tasks**. Here you can create a Gateway Task to send a project from the Controller to an Agent.

**Note:** The same actions applied for transferring resources between the **Developer, Test** and **Production** servers.

1. Click the **Create new Gateway Task** link. The EAM wizard will guide you through each step of the Send Project task.   
     
   Graphical user interface, text, application

   Description automatically generated
2. Not only will the Send Project send a project from the Controller to the Agent, but if the project already exists on the Agent, it will replace it.    
   Scroll down the list of Gateway Tasks and select **Send Project**. Press **Next**.   
     
   Graphical user interface, text, application, email

   Description automatically generated
3. Select **Execute Immediately**and press **Next**.
4. Select the **Agent** where you want to send your project and press**Next**.  
     
   Graphical user interface, application

   Description automatically generated
5. The EAM wizard knows your controller and agent configuration. It is going to prompt you to choose the machine where your project is located. In this example, the project that you want to send is located on the Controller or Gateway Source so select Local System and press Next.     
     
   Graphical user interface, text, application, email

   Description automatically generated
6. The EAM wizard will display all your projects from your local system. Select the project you want to send to your Agent and press Next.  (If you select your Agent on the previous screen, the EAM wizard will display all your projects running on your Agent that were sent over previously from the Controller).  
     
   Graphical user interface, text, application, email

   Description automatically generated
7. The EAM wizard summarizes all the information for you to review. Press Finish, and your project will be sent from the Controller to the Agent.  
     
   Graphical user interface, text, application

   Description automatically generated

The EAM wizard will execute the task, the screen will refresh, and a message will pop up stating that your Send Project task was successful.

1. To verify that your project was sent, go to your Agent, click on the Configure tab on the Gateway Webpage, and select Projects.   
     
   Graphical user interface, text, application, email

   Description automatically generated

**Note:** You can select to include inherited project resources.  The default is false.

Graphical user interface, text, application

Description automatically generated

**Send Project Resources Task**

You can also send project resources, such as project templates, windows, transaction groups, pipelines, and script modules from the Controller to an Agent.

1. On the **Controller**, under the **Configuration**section of the Gateway Webpage, go to **Enterprise Administration / Agent Tasks**. Create a Gateway Task to send project resources from the Controller to the Agent. (Same procedures for development to Test and Production servers.)
2. Click the **Create new Gateway Task** link. The EAM wizard will guide you through each step of the Send Project Resources task.
3. Click on the **Send Project Resources** radio button and press **Next.**

Graphical user interface, text, application, email

Description automatically generated

1. Schedule the Send Project Resources task to **Execute Immediately and** press **Next**.
2. Select the **Agent**on the Agent Selection window and press **Next**.
3. Select the Source Gateway / **Local System**where your project resources reside and press **Next.**
4. This example sends selected resources from the Training project on the Controller to the Agent. We selected **Project East.**Click **Next**.

Graphical user interface, text, application, email

Description automatically generated

1. You can see all the resources inside of the Training Project. It can include templates, windows, transaction groups, pipelines, and script modules. Choose whatever resources you want to send to the agent, press **Next**.

Graphical user interface, text, application

Description automatically generated

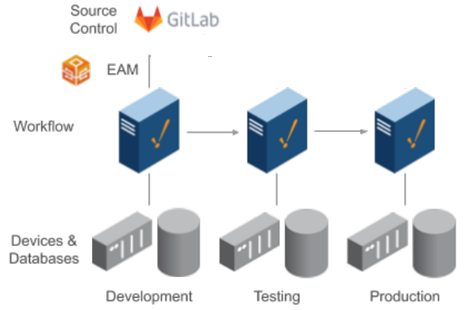
1. You have the option of either merging the project resources into an existing project or creating a new project on the Agent. Click **Next.**

Graphical user interface, text, application, Word

Description automatically generated

1. The EAM wizard summarizes all the information for you to review. Press **Finish.**
2. The Gateway Task for Send Project Resources will execute, the screen will refresh, and a message will appear stating that the task was successful.
3. To verify that the Training project resources were sent, go to your **Agent,**click on the **Configure**tab on the Gateway webpage, and**select Projects**.

## Step 2 – Enterprise Administration Module with Git tools

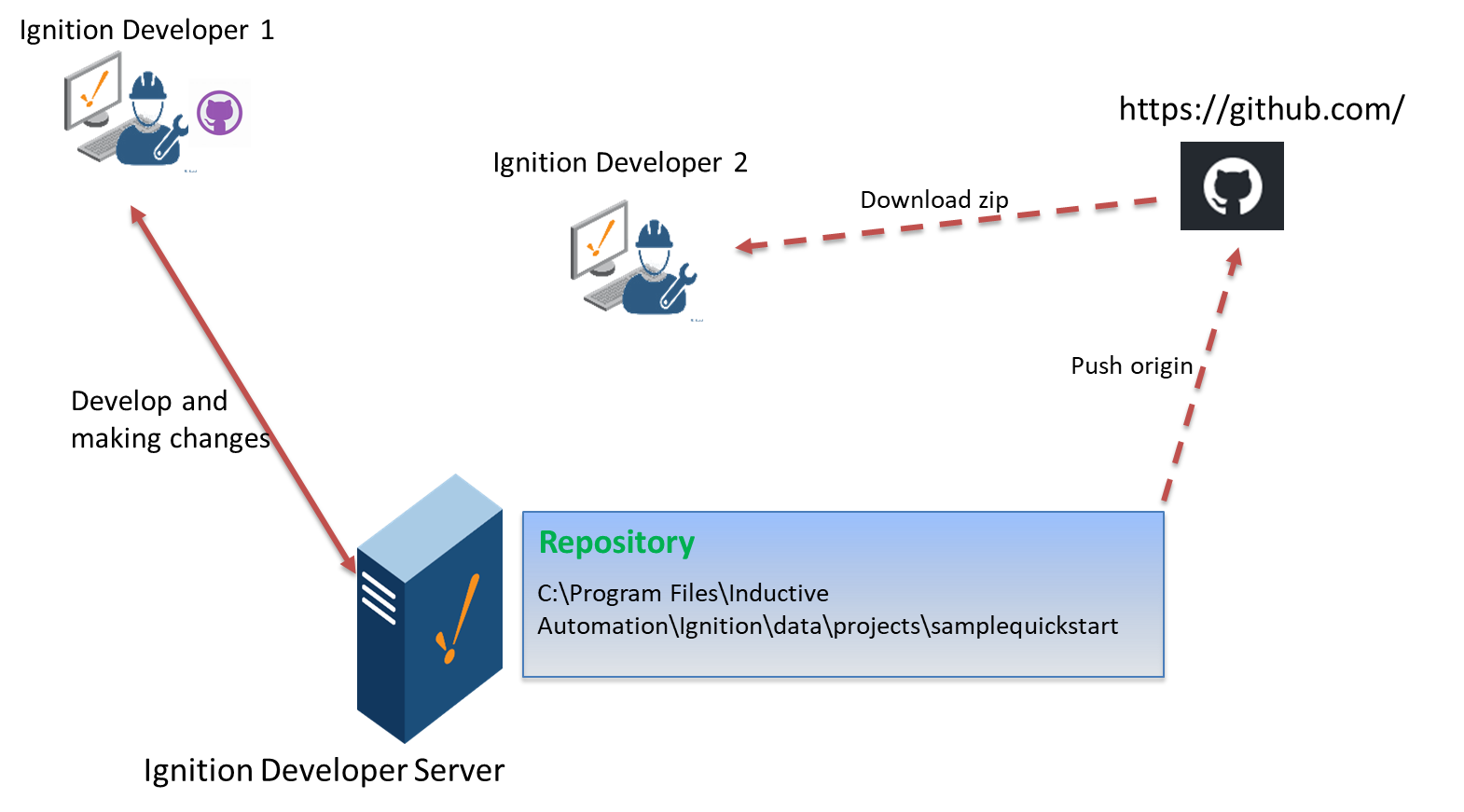


In **Step 1** we simply **backed up the projects and transfer files between Development**, Test and Production environment. That allows us to manage a disaster recovery plan in case of any failures, but no change management of any changes or added functionality.

In **Step 2** we will add **Software Change Management** to the Development environment to track any changes and gives us the ability to compare of fall back on different versions during the development phase. Software engineers can now work on the same resources and merging the changes into the development project before moving the resources to the Testing and Production using the EAM module.

We will use **GitHub** as our change management tool, but there are many other tools to choose from. It normally depends on your Company philosophy with considerations of brand, in the cloud or on premises resources. We will use the cloud ??????????????????????????????????????????????

See Section 6 to setup the GitHub accounts and structures.



### Start with a new project (Blank Project)

Use this section to start a repository if you have a blank or new project. The changes are good that you already have a project in your project’s directory. In that case, go to **Section 8.3.2**

* 1. Create a new Project in the Ignition Configuration. (**Config / Projects / Create new project…**)
  2. In **Name**, give it your project name. (AusCO)
  3. Click **Save**.
  4. Go to Github Desktop and create a new repository. (**File / New repository…**)
  5. Give it the same Name as your Ignition project name. (AusCO)
  6. Add a **Description** as an option and best practice.
  7. Click **Choose** and browse to the Project directory. (C:\Program Files\Inductive Automation\Ignition\data\projects)
  8. Click **Initialize this repository with a README.** You can use this file for further comments about the backup or release.
  9. Pick any **Git ignore** as an example or create one later.

Graphical user interface, application

Description automatically generated

* 1. Click **Create repository**.
  2. Click on **Publish Repository**.
  3. Now you can go into the Ignition Developer and start developing.

### Start with an existing Ignition project

Your Ignition project files are in: C:\Program Files\Inductive Automation\Ignition\data\projects\samplequickstart

Our aim is to make this our **new Repository.**

1. **Cut** (CTL + X) the AusCO directory from the projects and paste it somewhere save.
2. Click on **File / New Repository** in GitHub Desktop.
3. Make sure the **Name** is the same as your project name. (AusCO)

Give your repository a meaningful **Description**.

The **Local path** is: C:\Program Files\Inductive Automation\Ignition\data\projects

Graphical user interface, application

Description automatically generated

1. Click **Select Folder**

Graphical user interface, application

Description automatically generated

1. Click **Initialize this repository with a README.** You can use this file for further comments about the backup or release.
2. Pick any **Git ignore** as an example or create one later.
3. Click **Create repository**.
4. Copy all the files you saved back under the C:\Program Files\Inductive Automation\Ignition\data\projects\AusCO.
5. Edit and **Save** the **.ignore** file. (Delete everything and add *.resources/*. You don’t want that directory committed to the repository since it will be unique for each system.)

Text

Description automatically generated

You will notice all the new files inside GitHub Desktop.

Graphical user interface, text

Description automatically generated

1. Add a **short descriptive name** and **comments** at the bottom left inside the GitHub Desktop,

Graphical user interface, text, application, chat or text message

Description automatically generated

1. Click **Commit to main**

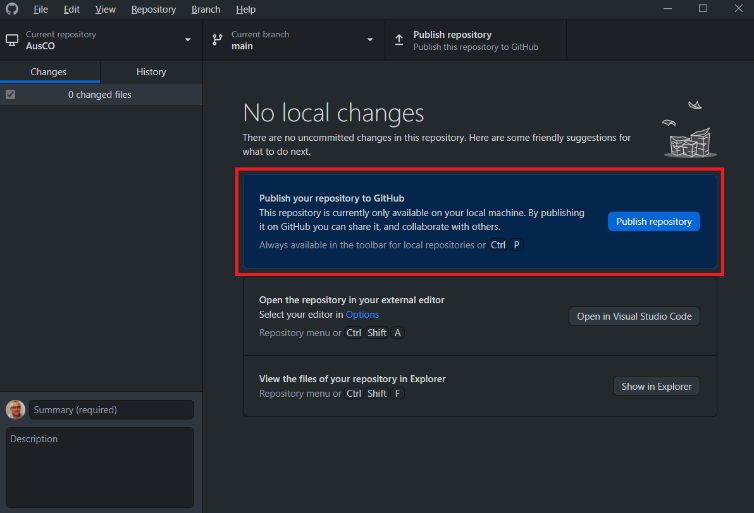
Your changes are now all in the **Staging area** in your **local repository** and GitHub Desktop keep track of any further changes. This is our **first step** towards Software Change Management, managing our changes.

Normally you work in a bigger team and not the only engineer working on the project. We can **share the resources** with other colleagues in the team.

Secondly, it is a good idea to **backup your work** and **store it in a different drive** or space. For that we will use the **Online GitHub repository**.

### Send my project resources to the Online GitHub Repository

On the GitHub Desktop, you will notice the “Blue area” indicating you have **local commits** that is on your local repository but not yet updated in the Online GitHub repository.



1. Click on **Push origin** to send an update with all the changes to the **Online GitHub**.

A popup appears with your first publishing to confirm and edit your Online Github name.

1. Give it a **Name**: AusCO

Meaningful **Description** as good practice.

1. Unselect **Keep this code private** if you plan to share the repository.
2. Click **Publish repository**.

Graphical user interface, text, application

Description automatically generated

### Proposed SCM procedure

Let us add some more meat to the procedure which will allow more control and information that will help tracking and manage changes.

1. Add e-mail information. See Appendix 10.3 to add your email.
2. Start by using **Issues** and track the change from the change **request** to **completion**.
3. Use branches for any changes.

#### Create an Issue

1. Go to your **GitHub Online** ( <https://github.com/Careldejager/AusCO> )
2. Click on **Issue**.
3. Complete the form and click **Submit new issue**.

Graphical user interface, text, application, email

Description automatically generated

#### Create a new branch

1. Always make sure you have the latest copy from the Online master before starting any new jobs. Click on **Fetch origin** to update your repository.
2. Click on **Current branch** in your GitHub Desktop. You are on the **main** branch.
3. Click on **New branch**.
4. Give the branch a **Name**. (We call it the same as the SR number)

Graphical user interface, text, application

Description automatically generated

1. Click **Create branch**.
2. Go to your Ignition developer and make the changes as requested in SR0001. Make sure you **save the changes** after your changes.

Chart, radar chart

Description automatically generated

Your changes will automatically reflex in GitHub Desktop.

Graphical user interface, text

Description automatically generated

1. Commit your changes. Click on **Commit to SR0001**.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. Now Publish your branch to GitHub Online. Click on **Publish branch**.

You should receive an e-mail notification from GitHub online confirming your action.

Graphical user interface, text, application, email

Description automatically generated

Note that the changes are NOT in the main branch and reside separately in the SR0001 branch.

GitHub Desktop opens the option to create a pull request to transfer the changes back to the main branch. That allows you to commit and collaborate your changes with the rest of the team.

Graphical user interface, text

Description automatically generated

1. Click Create **Pull Request**.
2. The Online GitHub opens up the page to comment and confirm the pull request. Add comment if needed.
3. Click on **Create pull request**.

Further options and information open.

Graphical user interface, text, application, email

Description automatically generated

1. Click **Close with comment**.

The pull is now completed and closed.

A picture containing graphical user interface

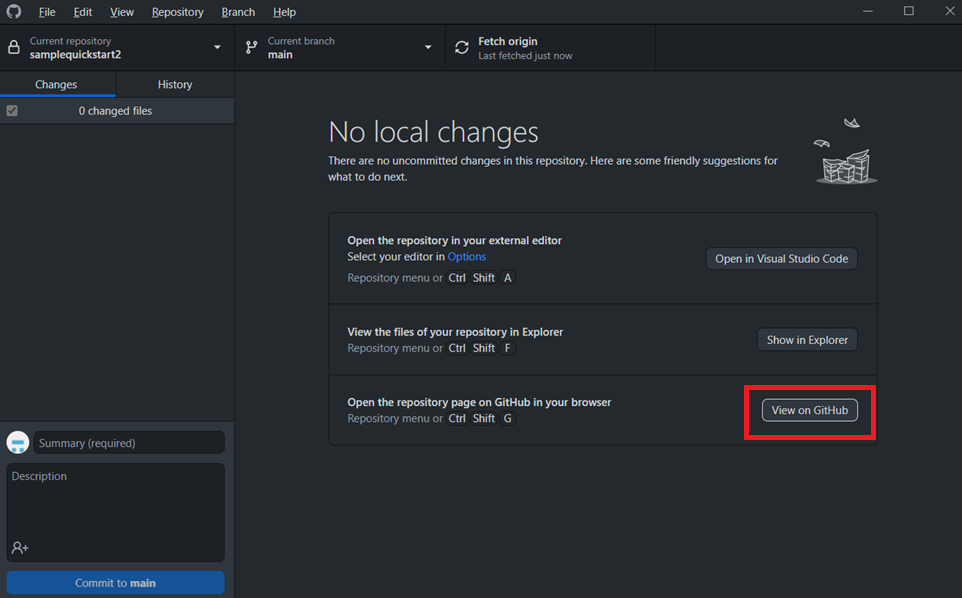
Description automatically generated

**Note:** **Ignition Tags, images** and any Ignition changes in the **configuration** browser are **NOT** part of the project but associated with the Ignition Gateway itself. Let us add those items as part of the backup in the online repository.

### GitHub Online (*https://github.com/)*

Let us look at the GitHub Online and confirm all our changes are online and the same as our local copy on the Ignition Gateway.

1. Click on **View in GitHub** on your GitHub Desktop application.



Your web depository opens in the web browser.

Table

Description automatically generated

1. Click on any of the **change requests** in the list. (Change request 2)

Notice the changes for each request

Graphical user interface, text, application

Description automatically generated

1. Click on **any number** in the file and type a comment.
2. Click **Add single comment**.

All these comments are transparent to your colleagues that shared the repository.

Graphical user interface, text, application

Description automatically generated

### Get a project backup from the GitHub Online

There are different ways to manage and transfer copies of the project. Let us look at the simplest way to get a copy from the online repository to your local repository.

1. From your main project page click on **Code**

A picture containing graphical user interface

Description automatically generated

You have different options:

* **Clone:**

By cloning with Git, you get the entire repository - all files, all branches, and all commits.

Cloning a repository is typically **only done once**, at the **beginning** of your interaction with a project. Once a repository already exists on a remote, like on GitHub, then you would clone that repository so you could interact with it locally. (Github, n.d.)

* **Open with GitHub Desktop**

It will open the repository in GitHub Desktop.

* **Download ZIP:**

Download the latest files in the repository in a zip format for easy distribution.

Graphical user interface, text, application

Description automatically generated

1. Click on **Download ZIP.**
2. You can distribute this copy to the relevant customers or colleagues.

Graphical user interface, text, application

Description automatically generated

## Step 3 – Fully Integrated Git Technology

# References

Auvesey. (n.d.). Versiondog Product Brochure.

*Change Management in Software Engineering*. (n.d.). Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/change-management-in-software-engineering/

git. (n.d.). *Getting Started - What is Git*. Retrieved from git: https://www.git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F

Inductive Automation. (2021, 06 30). Ignition 8. *Deployment Best Practices*.

Inductive Automation. (2022). *Gateway & Networks*. Retrieved from Ignition User Manual 8.1: https://docs.inductiveautomation.com/display/DOC81/Gateway+Network

Inductive Automation. (2022). Ignition Online Manual. Inductive Automation. Retrieved from https://docs.inductiveautomation.com/

MDT Autosave. (n.d.). What is MDT Autosave.

Preston-Werner, T. (n.d.). *Semantic Versioning 2.0.0*. Retrieved from https://semver.org/

# Appendix

## Install GitHub desktop and register online

Install GitHub Desktop from <https://desktop.github.com/>

For more information on the desktop version, go to: <https://docs.github.com/en/desktop/installing-and-configuring-github-desktop>

Go to File New Repository

Complete the Name: SampleQuickStart (No spaces in the name)

Description: My Sample Quick Start Project

Local Path: C:\Program Files\Inductive Automation\Ignition\data\projects (This is your project directory)

Select Initialise this repository with a README”

Git Ignore: ??????????????????????????????

Leave the rest default

(What if I already have a project)

Graphical user interface, text, application, email

Description automatically generated

Click Create repository (no write mkdr)

GitHub will automatically create directory

## Release Project Versions (Preston-Werner, n.d.)

## Add e-mail notifications

1. Open your Online Github account.
2. Go to Settings. Click on Email Notifications in the Integrations group.
3. You need to Confirm access. Click on Use GitHub Mobile or Use your password if you don’t have a Mobile app installed.
4. Provide your Password and Click Confirm.

The **Email notification** dialog box appear.

1. Add your email in the Address.
2. Add a meaningful Approved header description
3. Make sure Active is selected.
4. Click Setup notification.

Graphical user interface, text, application, email

Description automatically generated