# Developing as a Team

on the Now Platform

*Version 1.1 — Madrid*

## Introduction

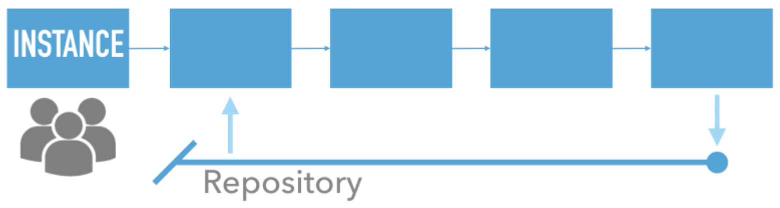
Enabling **simple yet powerful development team collaboration** is a core benefit of the Now Platform.

This document will help developers & teams adopt forward-looking, future-proof practices for team management, instance & application configuration, and source control usage. It is divided into 2 sections:

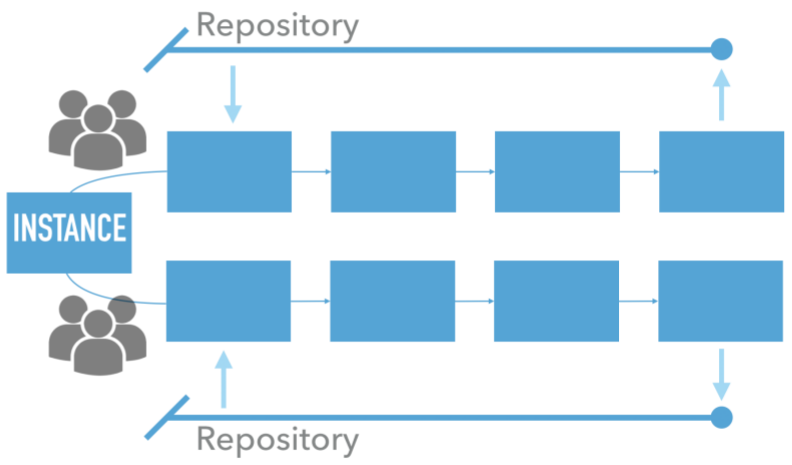
- Planning & preparation – this section will help identify the concepts, resources, and procedures you should familiarize yourself as you begin or evolve your ServiceNow application development.
- Step by step: implementing development team collaboration for an application – this section provides detailed how-to information for implementing modern, team-based development on the Now Platform. It also includes example developer workflows, training and educational links, and links to detailed walkthroughs in the appendixes.

This guide is intended to help your organization with 2 key scenarios:

1. Collaborate on a **single application** with multiple developers:



2. Collaborate on **multiple applications** with multiple developers:



## Planning & preparation

### Planning your application for development team collaboration

While the Now Platform supports a variety of approaches to application development, this guide will focus on the techniques that we recommend customers follow when creating new, forward-looking apps on ServiceNow. These guidelines will help make sure your application is easy to manage today, and that it aligns with future ServiceNow enhancements in application development, collaboration, management, and deployment.

Creating your applications as **Scoped Applications** and utilizing **Source Control Integration** will help align your organization for success with developer collaboration on the Now Platform.

To enable simple, future-proof collaboration for your app, we recommend that you utilize **Source Control Integration**. This will allow your organization to adopt industry standard toolsets while following modern application development & team management paradigms. See Appendix A for additional discussion, including a note about update sets. See the ServiceNow documentation for more on Source Control Integration.

ServiceNow also recommends that you create your apps as **Scoped Applications**. Global applications are not recommended for most use cases and are currently not supported with source control integration. You can read more about Application Scope in the ServiceNow documentation site.

## Setting up your source control repository

ServiceNow's Source Control Integration supports industry-standard git repositories. These can be hosted by a variety of vendors (Github, Gitlab, Bitbucket, etc.) or self-hosted.

Whatever repository host you use, **it will need to be accessible** from your ServiceNow instance(s). For public git hosting services, no additional work will be needed. If you are managing your own repository inside your corporate network, you will need to enable git API access from your ServiceNow instances, typically by mapping the correct external ports from your firewall or gateway to your internal repository.

Your ServiceNow instances will store the access credentials for your application repositories. Rather than using the passwords for your git repository accounts, ServiceNow recommends that you use personal access tokens to manage access to your repositories on a per-instance or per-application basis (with a dedicated token per-application or per-instance, as desired). This will allow you to utilize external source control accounts within your ServiceNow instances without sharing your actual 3rd-party passwords.

## Planning your team(s)

1. Defining Roles

   When setting up your development teams, you should have a few key roles in mind. Each of those roles should be assigned the minimum necessary permissions for performing their required tasks, so that users who should not be using restricted features (for reasons of security, ability, knowledge, or simple separation of responsibilities) do not have access to them. That said, in a smaller organization or team, one person may be filling several of these roles.

   For example, you might have separate roles for each of these, listed here in the order of least- to most-permissive:

   - Citizen Developer
   - Pro-code development
   - ATF test creation/admin
   - Application repo submission
   - User administration
   - Platform owner / architect

   A Citizen Developer only needs, and should only be assigned, access to no-code interfaces (Flow Designer, Workflow, etc., but no scripting) & ATF capabilities (so they can test their own work) on dev instances, while a delegated app administrator role would have access to create roles & assign permissions for other users – a capability that other developers should not be assigned.

For a detailed explanation of the various app-level permissions you will need to set for each role, see the Delegated development & deployment documentation.

2. Training

   Each team member should be familiar with the appropriate level of application development for their role. ServiceNow's documentation for creating applications is a great place to start, with detailed guides and starter videos.

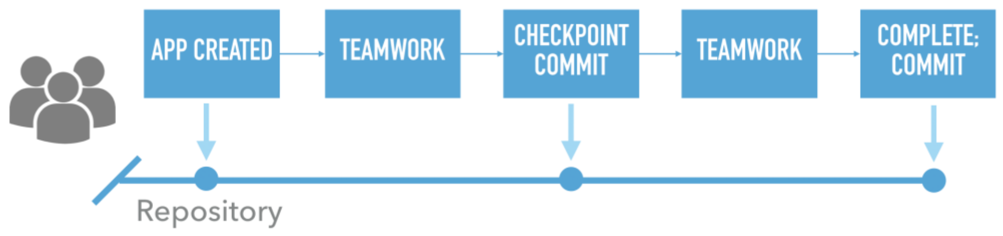
   ServiceNow's **Training and Certification Organization** recommends the following courses on the **Application Developer Learning Paths** for No-Code Citizen Developers and Low/Pro-Code Developers:
   PDF Link

   ServiceNow's **Developer Portal** also provides a Build My First Application self-paced offering to help you get familiar with application development.

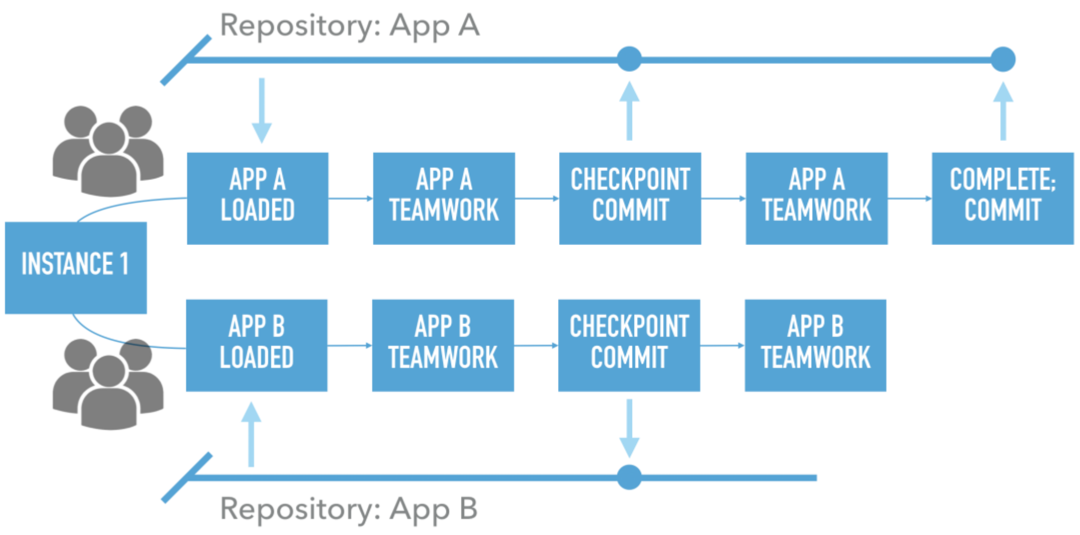
   Select the offering that works best for you.

## Planning your ServiceNow instances

The simplest way to collaborate on an application is with developers **working together on a single ServiceNow instance**. Generally speaking, these developers should all be working on the same task or update, with the expectation that the application will be committed once the task or update has been finished or reached a stable checkpoint:



Multiple teams can share a single instance as well, with each team working on a different app:



For larger teams, multiple instances can enable work on multiple, separate tasks simultaneously for a given app, with each task following its own development & commit timeline. For additional guidance on developer flows utilizing these instance configurations, see the example sections for single & multiple apps.

## Planning to test

The single biggest recommendation we can make regarding testing is to make sure you test. Testing is at its most effective when you plan for it from the start and test as you complete each part of your application during development.

ServiceNow provides extensive automated testing support via the Automated Test Framework (ATF), and manual testing via the Test Management tool.

There are a few ATF features that we recommend you utilize for any new app:

- Parameterized tests: these allow you to create a test and the test data separately. This way, you can create a test and then run it against multiple different sets of data, while also making it much easier to update your tests & add or modify data sets over time, independently of one another.
- Scheduled suites: guarantees that tests are run and to help maintain quality standards over time.
- Custom UI testing: used for UI elements you create, including Modals, UI pages, Service Portal Widgets, Interceptors, and Wizards

## Deployment & distribution planning

When working on an application that you want to distribute to other instances (e.g. for additional development or testing on other dev & sandbox / test instances), the simplest method is to commit it. The application will then be available to those instances via **source control**.

Source control is an appropriate distribution mechanism when you are distributing an application in order to continue working on it (for example, to enable other teams or team members to contribute). For testing your application on dev instances, either source control or the **Application Repository (App Repo)** could be used to distribute your application, depending on what stage of testing you are in.

When you are close to releasing an application to your users, you should use the Application Repository to load the application onto your QA or Test instances for testing, so that the build you are testing more closely matches the final deployed version of the app.

When distributing an application to a Production instance, **you should always use the App Repo**.

**Documentation**: When to use the application repository, Install an application, Import application from source control

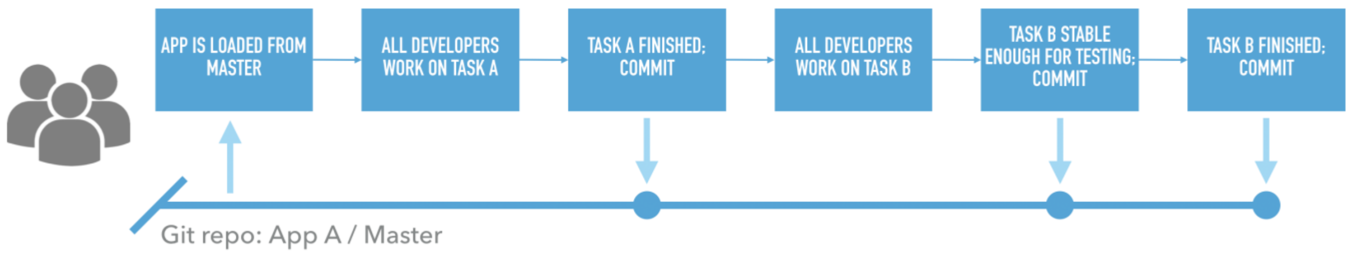
## Step by step: implementing development team collaboration for an app

This section assumes you have access to a ServiceNow instance (London or later release version), and you should be logged-in as admin. If you don't have a ServiceNow instance available, you can follow the Build My First Application tutorial for instructions on getting a Personal Development Instance.

### Development & coordination within a single app

If you have multiple developers working on the same application on the same instance, you should plan your commits for when all current work (across all developers) is finished or at a stage where it can be safely committed (e.g., if you have several developers working on a single task to update your app, you should commit once the application reaches a stable checkpoint or is complete).

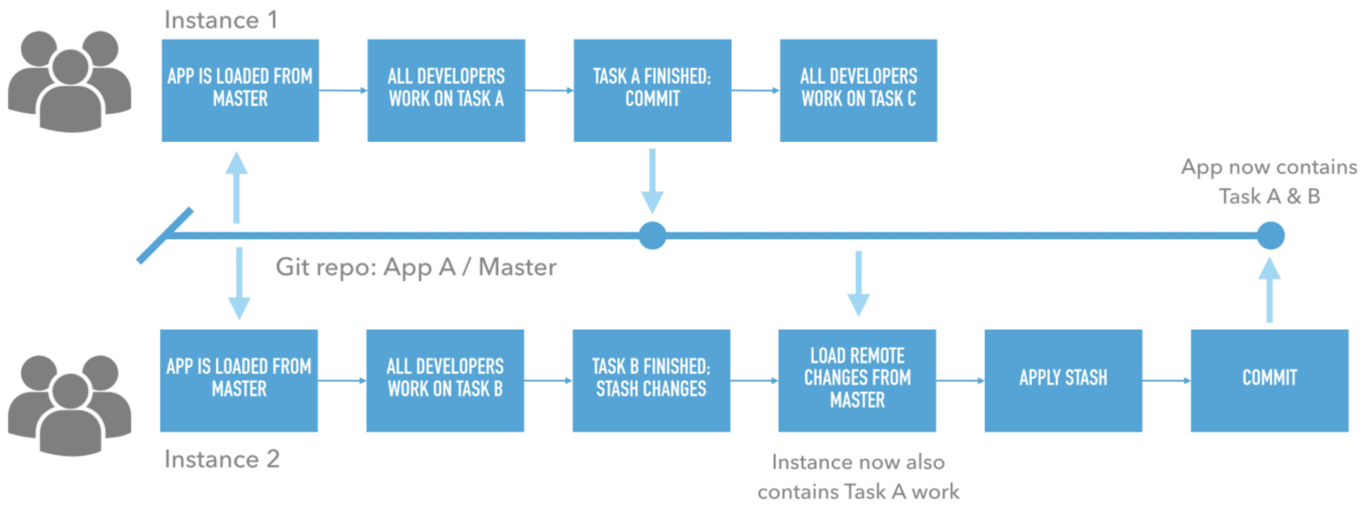
Example dev flow (single instance):


APP IS LOADED FROM MASTER
ALL DEVELOPERS WORK ON TASK A
TASK A FINISHED; COMMIT
ALL DEVELOPERS WORK ON TASK B
TASK B STABLE ENOUGH FOR TESTING; COMMIT
TASK B FINISHED; COMMIT
Git repo: App A / Master

As of the London release, committing or stashing will result in all current changes to an application being committed or stashed. Selective committing & stashing is being considered for a future release.

To coordinate work on the same application across multiple instances, developers should plan to coordinate their commits. After the first instance has committed, the second instance should stash their changes, reload the application from source, and re-apply the stash.

Example dev flow (multiple instances):

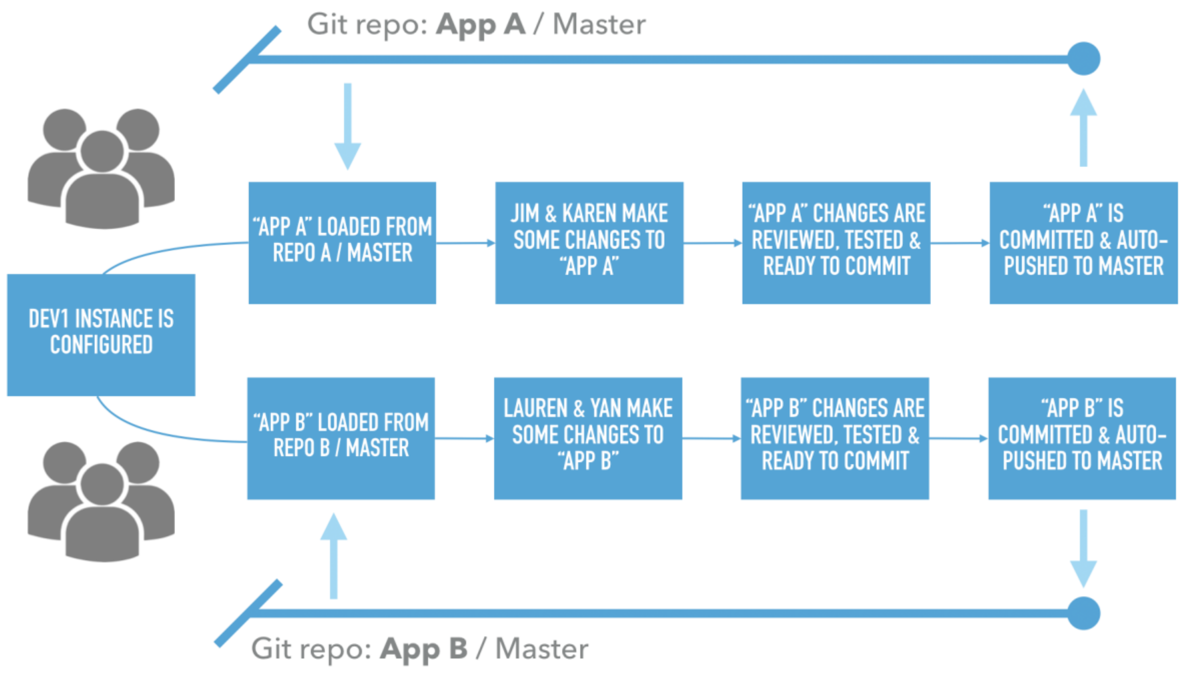

Instance 1
APP IS LOADED FROM MASTER
ALL DEVELOPERS WORK ON TASK A
TASK A FINISHED: COMMIT
ALL DEVELOPERS WORK ON TASK C
App now contains Task A & B
Git repo: App A / Master
APP IS LOADED FROM MASTER
ALL DEVELOPERS WORK ON TASK B
TASK B FINISHED: STASH CHANGES
LOAD REMOTE CHANGES FROM MASTER
APPLY STASH
COMMIT
Instance 2
Instance now also contains Task A work

To minimize conflicts in this case, it's best to avoid editing the same files or records on both instances at the same time; merging file changes or otherwise dealing with conflicts must be performed manually when applying a stash, by selecting "Manually Apply" on the Apply Stashed Changes conflict alert screen. Merge support is being considered for a future release.

## Development & coordination across multiple apps

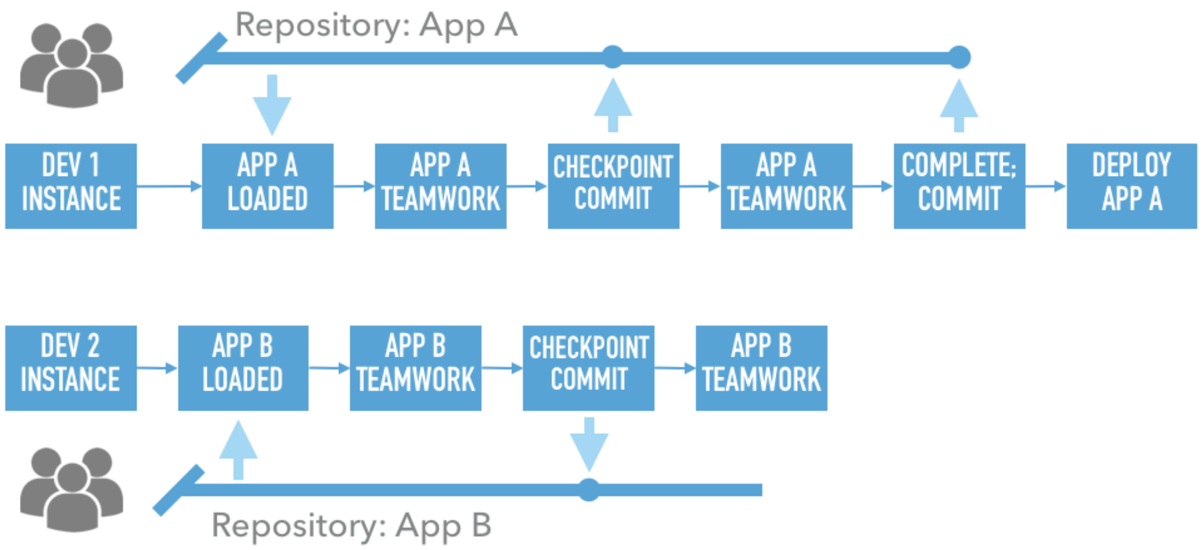
The developer workflow for multiple teams working on multiple apps on the same instance is relatively straightforward, as there is no conflict and no coordination required between application teams:  each application has its own source code repository. Once each developer selects their app in the ServiceNow Studio, their development workflow is identical to the previous section on development collaboration for a single app.

## Example dev flow (multiple apps):



Git repo: App A / Master
"APP A" LOADED FROM REPO A / MASTER
JIM & KAREN MAKE SOME CHANGES TO "APP A"
"APP A" CHANGES ARE REVIEWED, TESTED & READY TO COMMIT
"APP A" IS COMMITTED & AUTO-PUSHED TO MASTER
DEV1 INSTANCE IS CONFIGURED
"APP B" LOADED FROM REPO B / MASTER
LAUREN & YAN MAKE SOME CHANGES TO "APP B"
"APP B" CHANGES ARE REVIEWED, TESTED & READY TO COMMIT
"APP B" IS COMMITTED & AUTO-PUSHED TO MASTER
Git repo: App B / Master

This developer workflow can also be implemented without change across multiple instances:



## Creating an application

The first step in collaborating on an application is to create the application; while there are detailed instructions on application creation available in ServiceNow's documentation site, additional resources include:

Walkthrough -- Appendix B: Creating an application

ServiceNow Training: Build My First Application

## Linking the application to Source Control

Before your team(s) can start collaborating on your application, you need to configure it so that it is stored in source control, and so that all future changes can be tracked within that source control repository.

Walkthrough -- Appendix C: Linking your application to source control

Documentation: Source control integration

## Adding team members & delegates

To enable your developers access to work on and deploy your application, you will need to assign them the correct roles.

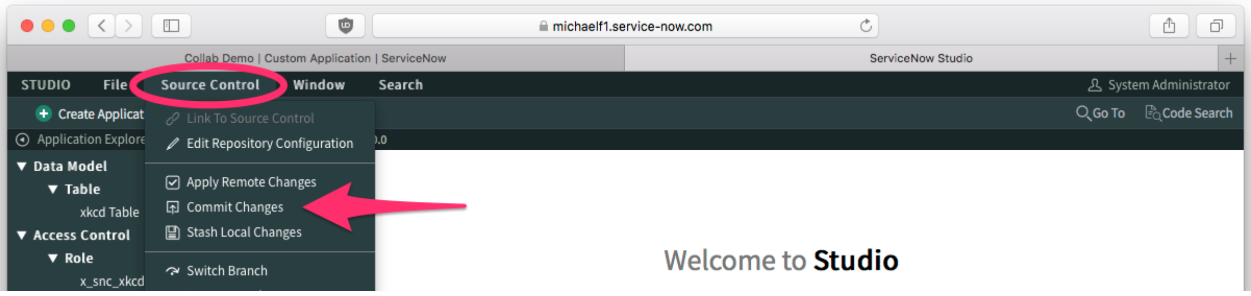
Walkthrough -- Appendix D: Adding team members and delegates

Documentation: Delegated development and deployment

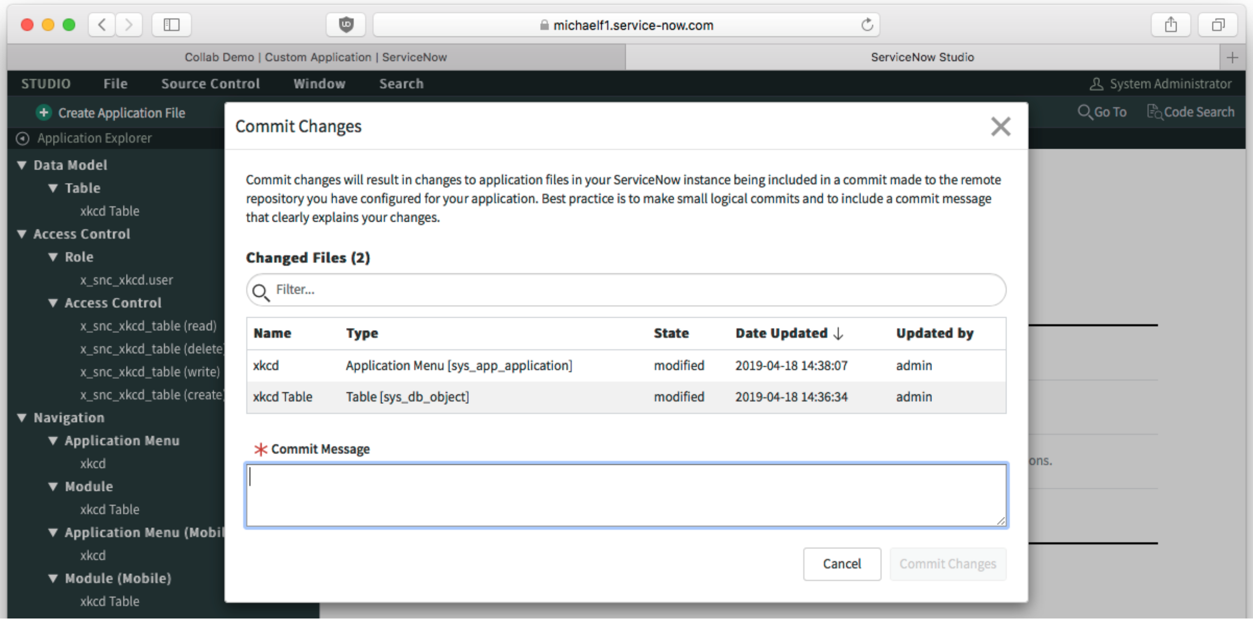
## Committing changes

As your developers are making changes to the app, the system keeps track of those changes. At any point, you can review the current local changes and either stash or commit them (committing results in those changes being immediately pushed to the branch as well).

To review your current uncommitted changes, click on **Source Control** and then either **Commit changes** or **Stash Local Changes**:



This presents the list of current changes:



Commit changes will result in changes to application files in your ServiceNow instance being included in a commit made to the remote repository you have configured for your application. Best practice is to make small logical commits and to include a commit message that clearly explains your changes.

Changed Files (2)

| Name | Type | State | Date Updated | Updated by |
|---|---|---|---|---|
| xkcd | Application Menu [sys_app_application] | modified | 2019-04-18 14:38:07 | admin |
| xkcd Table | Table [sys_db_object] | modified | 2019-04-18 14:36:34 | admin |

From there, you can either commit (adding a message for the commit) or stash the changes.

Documentation: Commit changes, Stash local changes, Available source control operations

## Testing your app

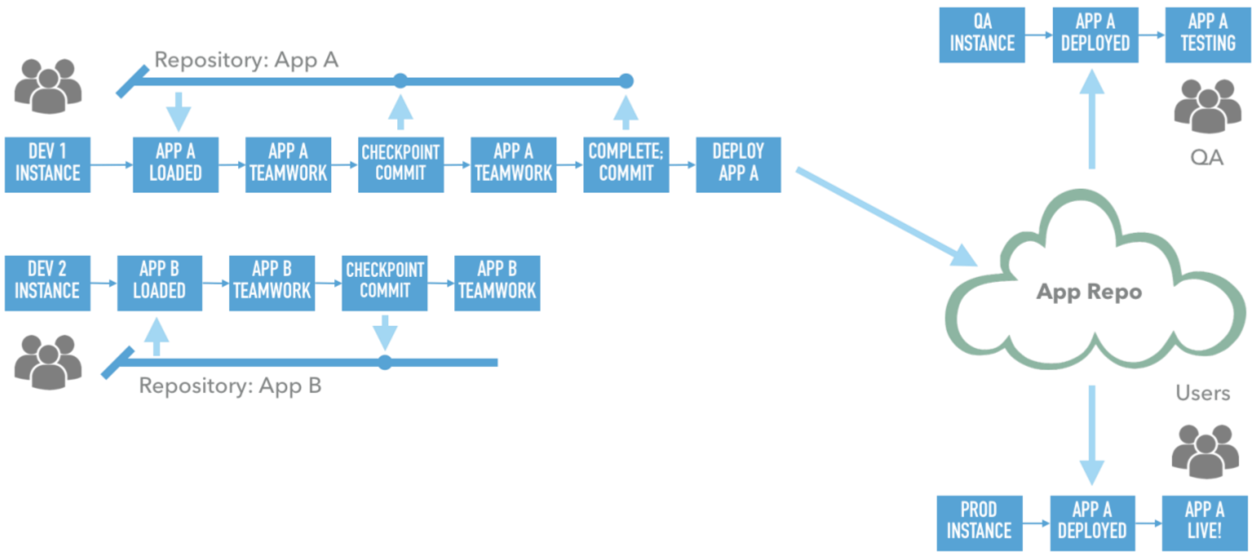
Prior to deploying any apps or application updates to the App Repo, they should be fully tested. ServiceNow's Automated Test Framework and Test Management features will help insure that the application works as designed.

servicenow

Documentation: Automated Test Framework, Test Management

## Publishing your app

Publishing an application to the App Repo makes this version of the application available to all of your organization's ServiceNow instances. This is how your app should be deployed to your QA / Test instances (for testing) and finally to your Production (Prod) instances, for your users.



Documentation: Publishing an application to the application repo, Install an application

## Conclusion

The Now Platform is an ideal environment for modern, collaborative development. With source control integration, rich team management facilities, robust test & deployment options, and extensive training, documentation, & support resources, ServiceNow empowers your teams to create applications that will help drive success across your organization. **We can't wait to see what you create!**

If you would like to leave any feedback regarding this document, please email platformenablement@servicenow.com and let us know what you think.

## Appendix A: Source Control vs Update Sets

### Source Control: current & future

While source control (example: git) is a great way to align your organization with current and future best-practice trends in ServiceNow application development & management, there are some reasons that an organization might not be ready to adopt the current source control integration:

- **Branch merging** is not currently supported
- Corporate git repositories **behind a firewall** cannot be directly integrated without additional firewall configuration
- **Global applications** are not currently supported

Each of these is being considered as a future enhancement by ServiceNow.

In cases where support for these use-cases is essential, the Now Platform enables the use of Update Sets and related features to provide similar capabilities (with caveats and notable precautions required – see below).

### Update Sets: Best Practices

As the saying goes, "with great power comes great responsibility". Update Sets provide deep-reaching access into the capabilities of the platform, but they can also pose risks, and responsible usage is advised.

Because update sets make direct changes to a live instance, you should review "Get started with update sets" to avoid errors, performance issues, and **changes that could impact upgradeability**.

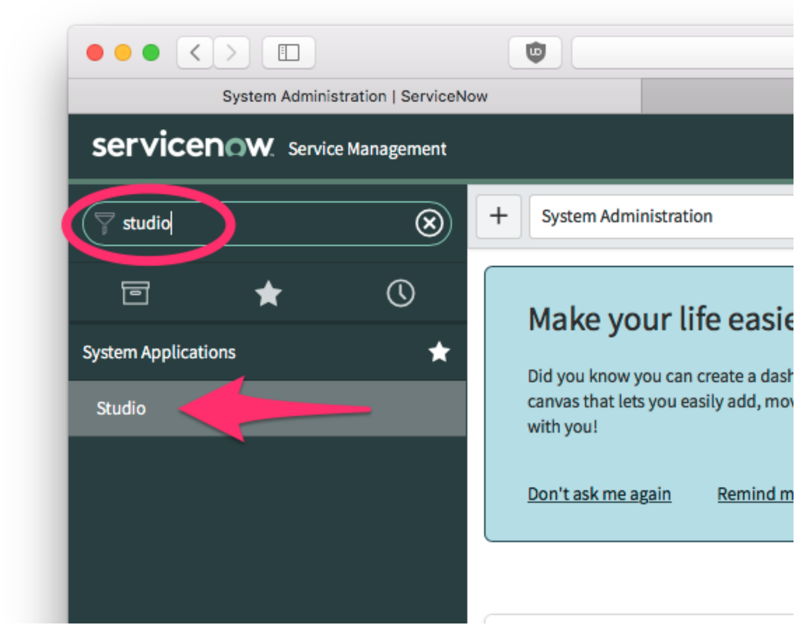
ServiceNow's full documentation on System Update Sets should also be reviewed to help you learn how to plan the update process and avoid other common mistakes.

**Documentation**: System Update Sets

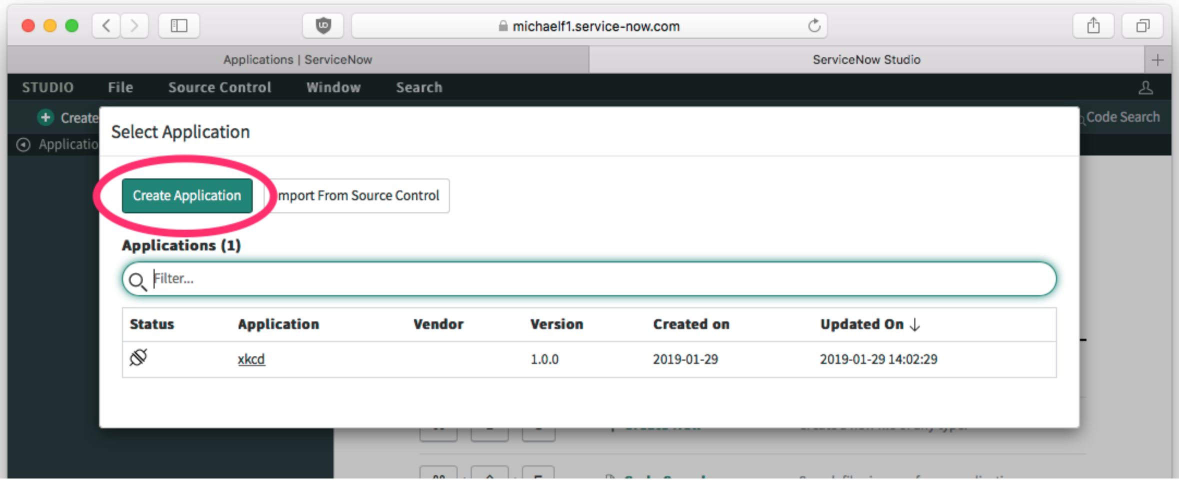
## Appendix B: Creating an application

Here's a simple example of how to get started with an empty application:

Launch the ServiceNow Studio:


System Administration | ServiceNow
servicenow Service Management
studio
System Administration
System Applications
Studio
Make your life easie
Did you know you can create a dash
canvas that lets you easily add, mov
with you!
Don't ask me again
Remind m

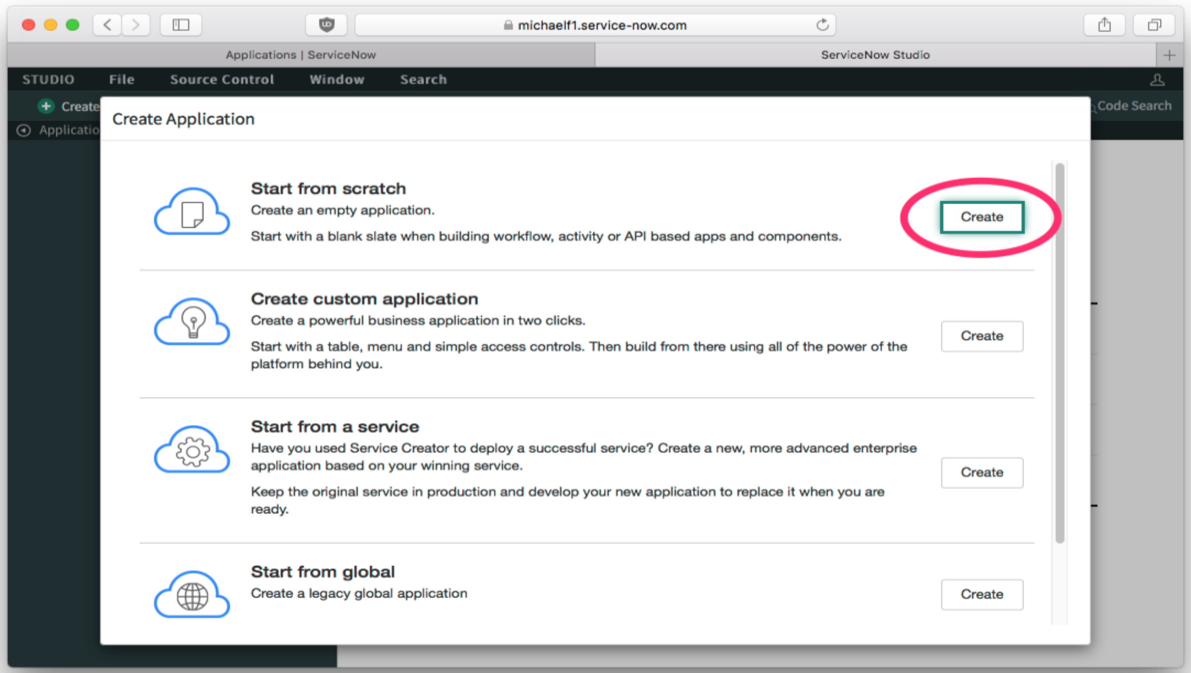
Then click on Create Application:



| Status | Application | Vendor | Version | Created on | Updated On ↓ |
|---|---|---|---|---|---|
| | xkcd | | 1.0.0 | 2019-01-29 | 2019-01-29 14:02:29 |

Side note: see the "Import from Source Control" button? That's one way we can load this application on other instances later.

For this example, we'll select start from scratch:



Create Application

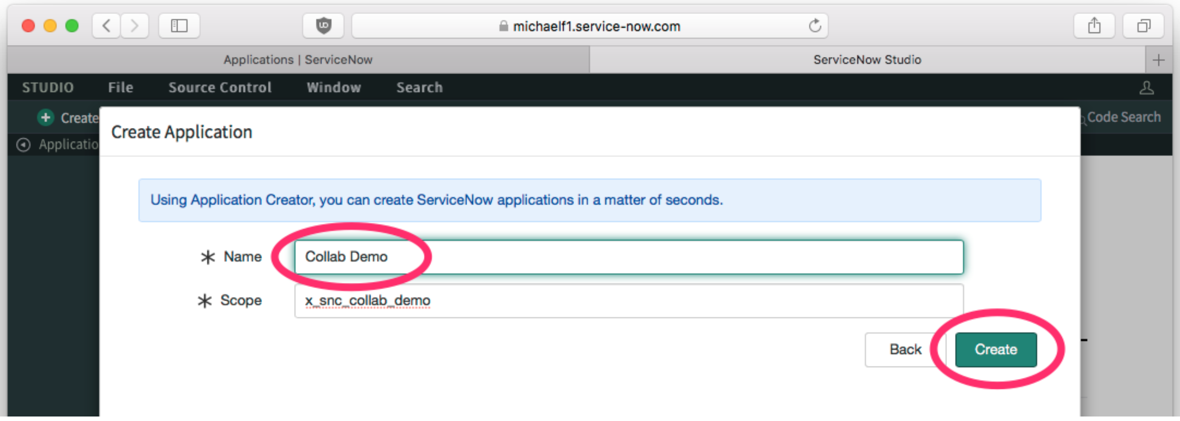
**Start from scratch**
Create an empty application.
Start with a blank slate when building workflow, activity or API based apps and components.
Create

**Create custom application**
Create a powerful business application in two clicks.
Start with a table, menu and simple access controls. Then build from there using all of the power of the platform behind you.
Create

**Start from a service**
Have you used Service Creator to deploy a successful service? Create a new, more advanced enterprise application based on your winning service.
Keep the original service in production and develop your new application to replace it when you are ready.
Create

**Start from global**
Create a legacy global application
Create

Give your application a name and click **Create**:



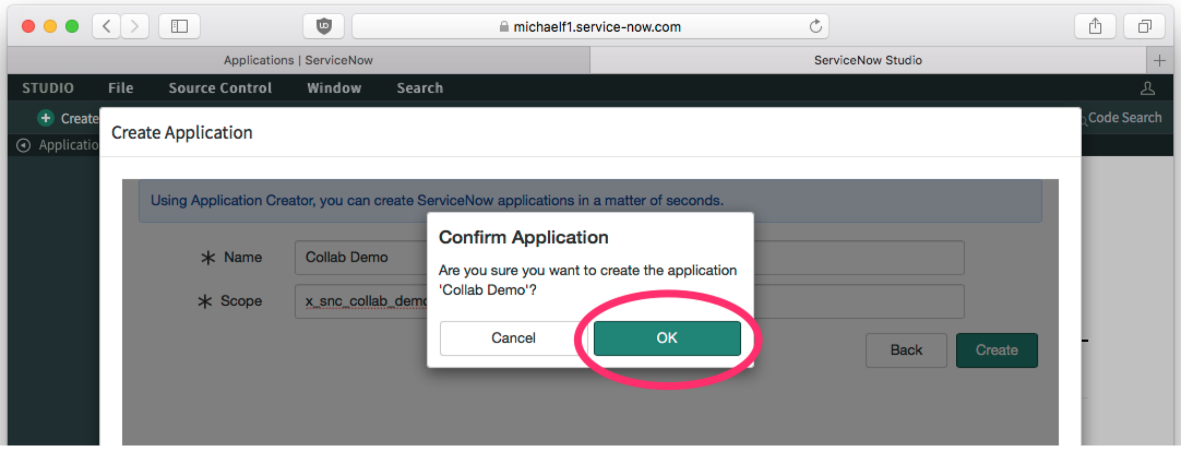
Create Application

Using Application Creator, you can create ServiceNow applications in a matter of seconds.

| | |
|---|---|
| * Name | Collab Demo |
| * Scope | x_snc_collab_demo |

Back　Create

Confirm:



Create Application

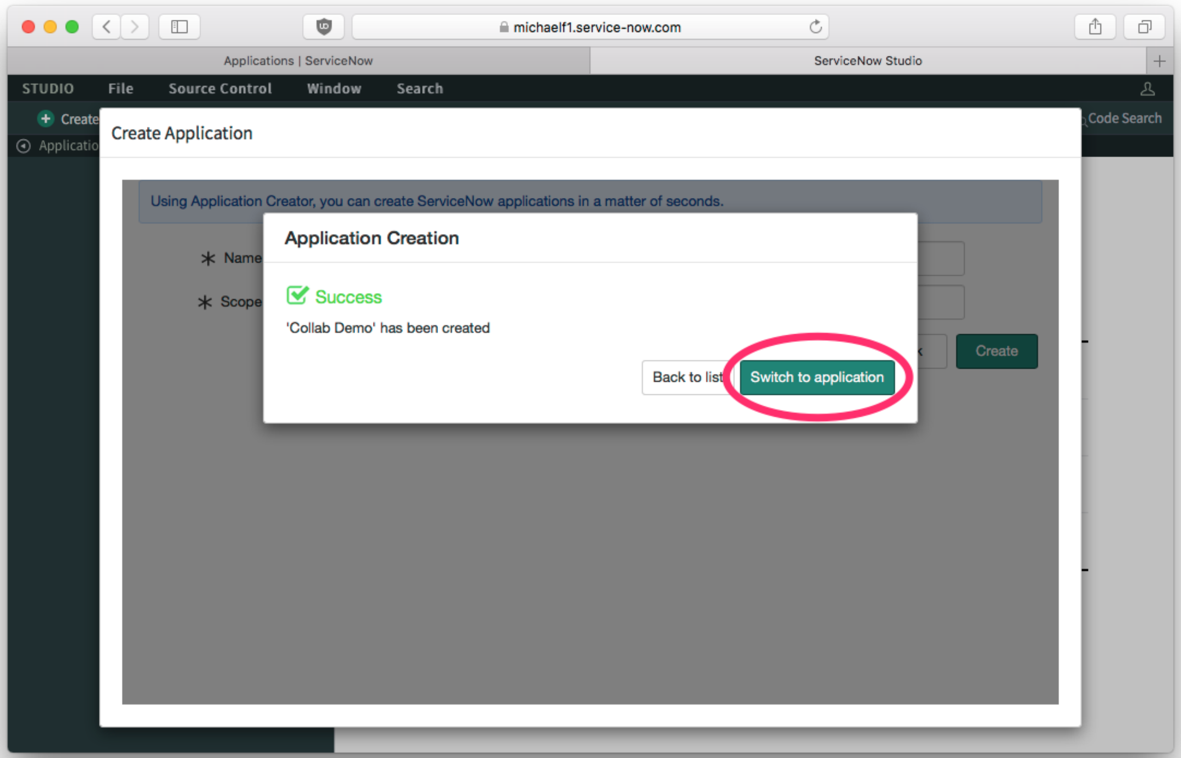
Using Application Creator, you can create ServiceNow applications in a matter of seconds.

* Name: Collab Demo
* Scope: x_snc_collab_dem

Confirm Application

Are you sure you want to create the application 'Collab Demo'?

Success -- now click **Switch to application** to load our new application in the Studio.



Create Application

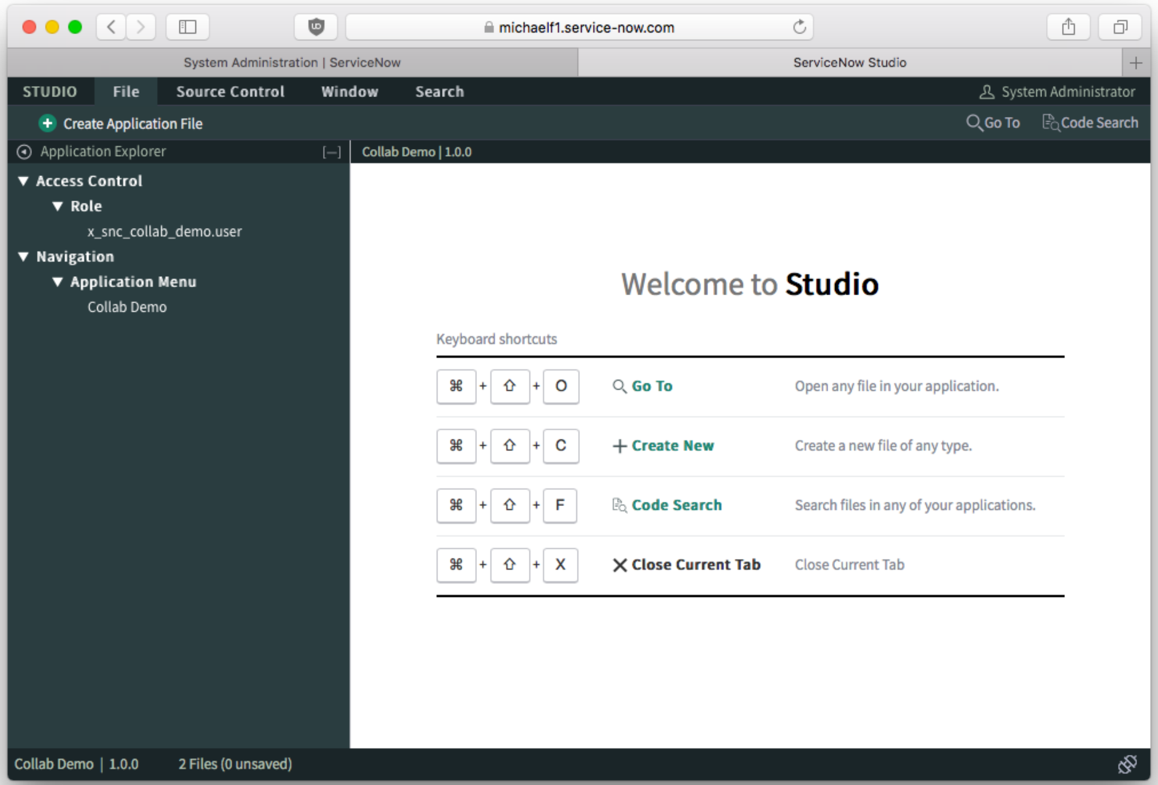
Using Application Creator, you can create ServiceNow applications in a matter of seconds.

Application Creation

Success

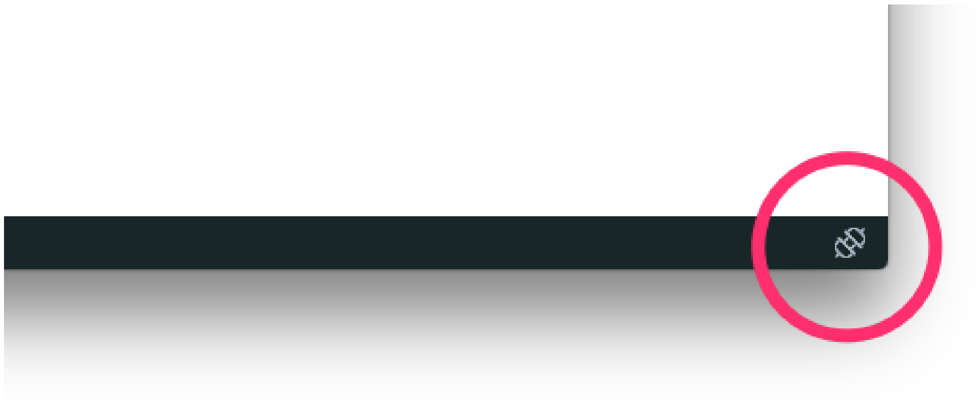
'Collab Demo' has been created

That's it! Here's our app:



| | | |
|---|---|---|
| ⌘ + ⇧ + O | Go To | Open any file in your application. |
| ⌘ + ⇧ + C | Create New | Create a new file of any type. |
| ⌘ + ⇧ + F | Code Search | Search files in any of your applications. |
| ⌘ + ⇧ + X | Close Current Tab | Close Current Tab |

… you may have noticed this little icon on the bottom right:



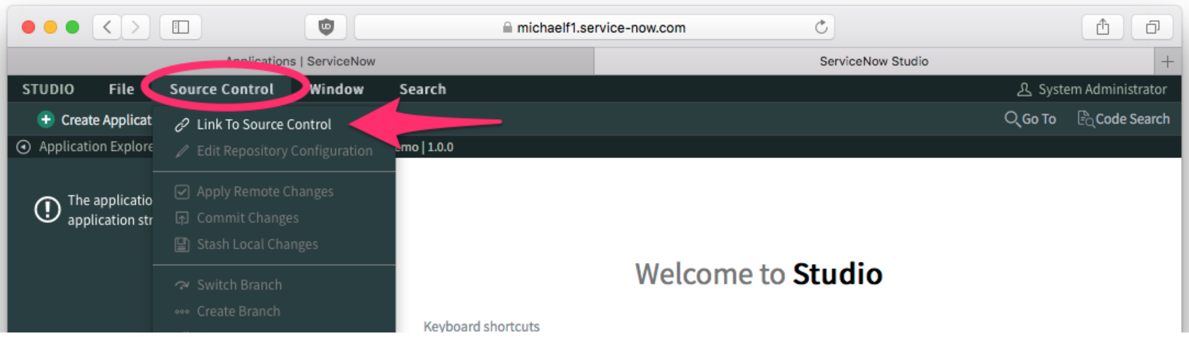
This means that our application is not yet connected to Source Control. We'll fix that in Appendix C: Linking your app to Source Control.

Documentation: Creating applications

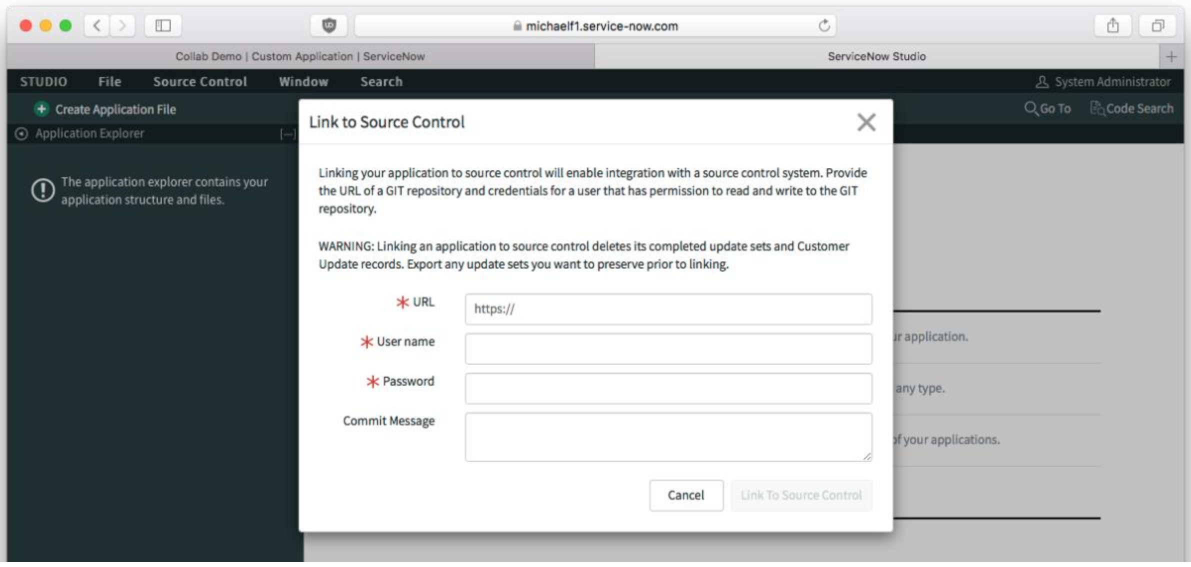
## Appendix C: Linking your app to source control

While detailed instructions on ServiceNow's Source Control Integration are available in the Service Docs site, here's a quick intro:

- First, click on the **Source Control tab**, and select **Link To Source Control**:



- Then enter your **git repo** details – including the URL, username, and password (we recommend using a personal access token):



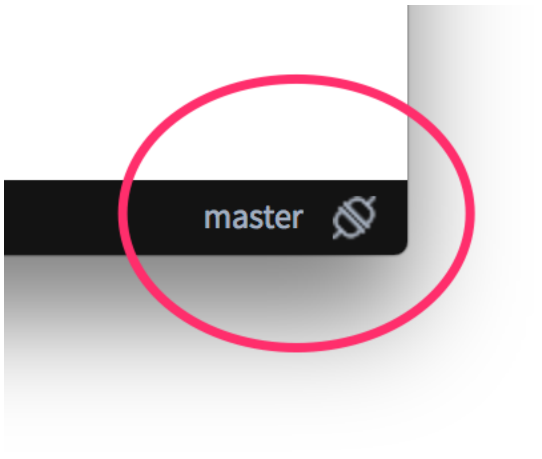
Linking your application to source control will enable integration with a source control system. Provide the URL of a GIT repository and credentials for a user that has permission to read and write to the GIT repository.

WARNING: Linking an application to source control deletes its completed update sets and Customer Update records. Export any update sets you want to preserve prior to linking.

|  |  |
|---|---|
| * URL | https:// |
| * User name |  |
| * Password |  |
| Commit Message |  |

Cancel Link To Source Control

- Once you have done this, the **Source Control icon** is updated to show the current state:



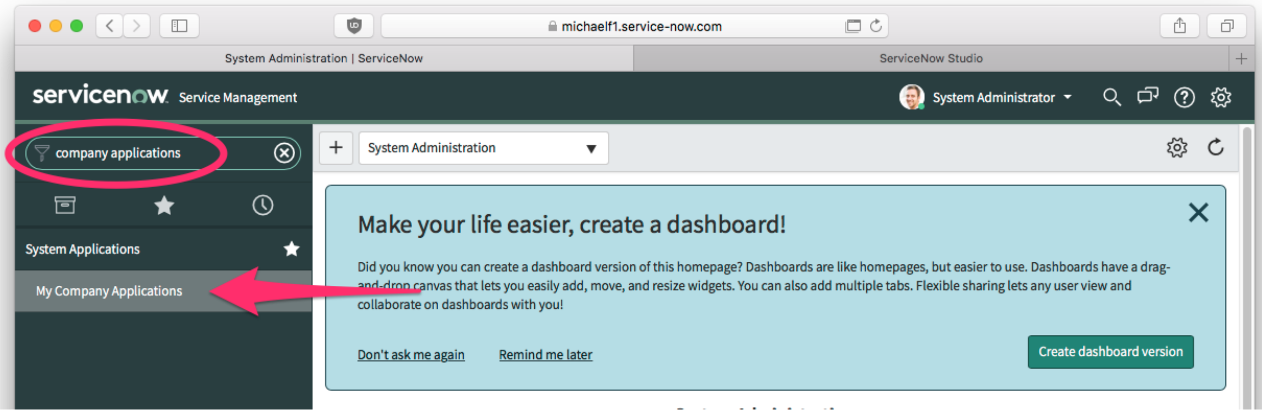
From here on, **all changes made to the application on this instance** will be visible in the relevant sections of the Source Control menu; actions in that menu will let you handle those changes (commit your instance's local changes to source control, stash local changes to source control so you can continue working on them later, etc.) as well as apply remote changes and other options. The Source Control icon will update to reflect the current state of the application (no changes, local changes, etc.)

For a list of all currently-available **Source Control actions**, please see page 1 of the Source Control Integration documentation linked below.

**Documentation**: Source Control Integration

# Appendix D: Adding team members & delegates

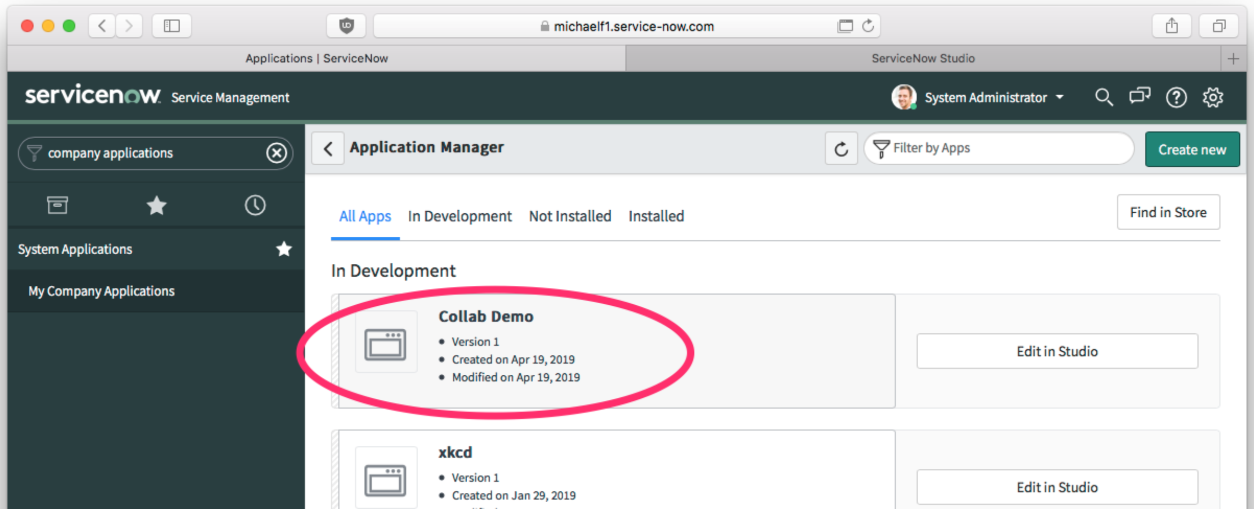
First, navigate to My Company Applications:



Make your life easier, create a dashboard!

Did you know you can create a dashboard version of this homepage? Dashboards are like homepages, but easier to use. Dashboards have a drag-and-drop canvas that lets you easily add, move, and resize widgets. You can also add multiple tabs. Flexible sharing lets any user view and collaborate on dashboards with you!

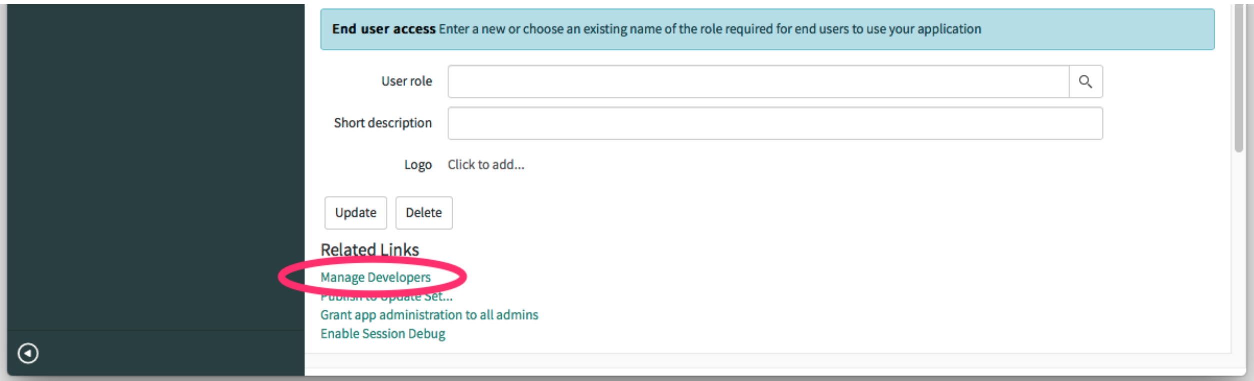
Next, click on the application name:



Collab Demo
- Version 1
- Created on Apr 19, 2019
- Modified on Apr 19, 2019

xkcd
- Version 1
- Created on Jan 29, 2019

In Related Links towards the bottom of the page, click Manage Developers:


**End user access** Enter a new or choose an existing name of the role required for end users to use your application

User role

Short description

Logo Click to add...

Update Delete

Related Links

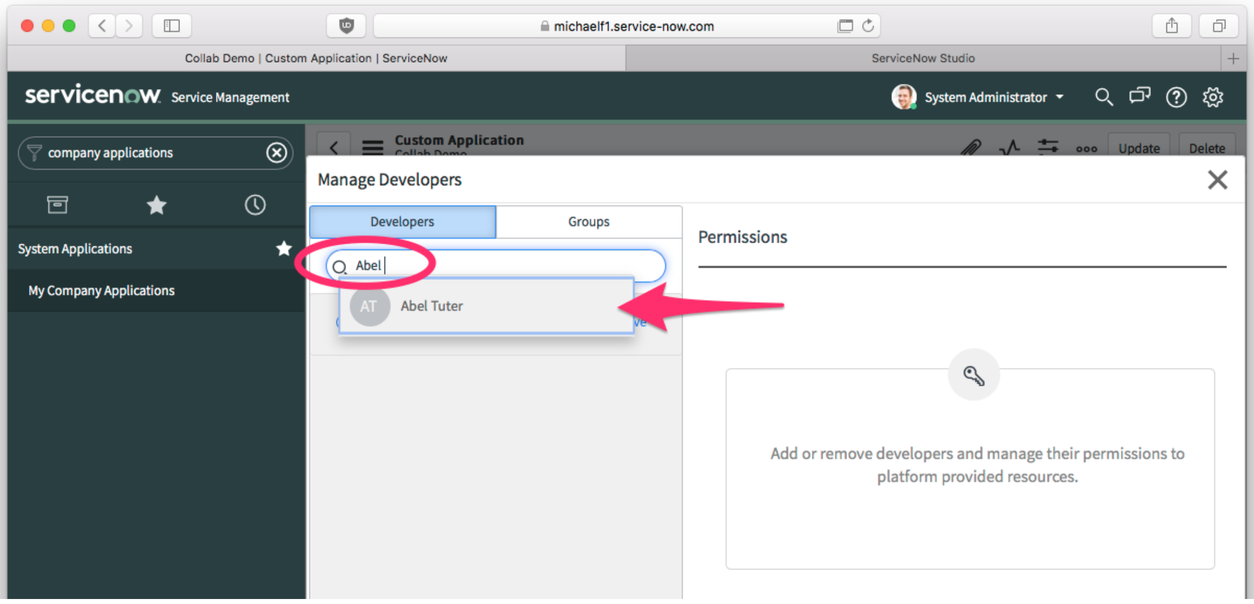
Manage Developers

Publish to Update Set...

Grant app administration to all admins

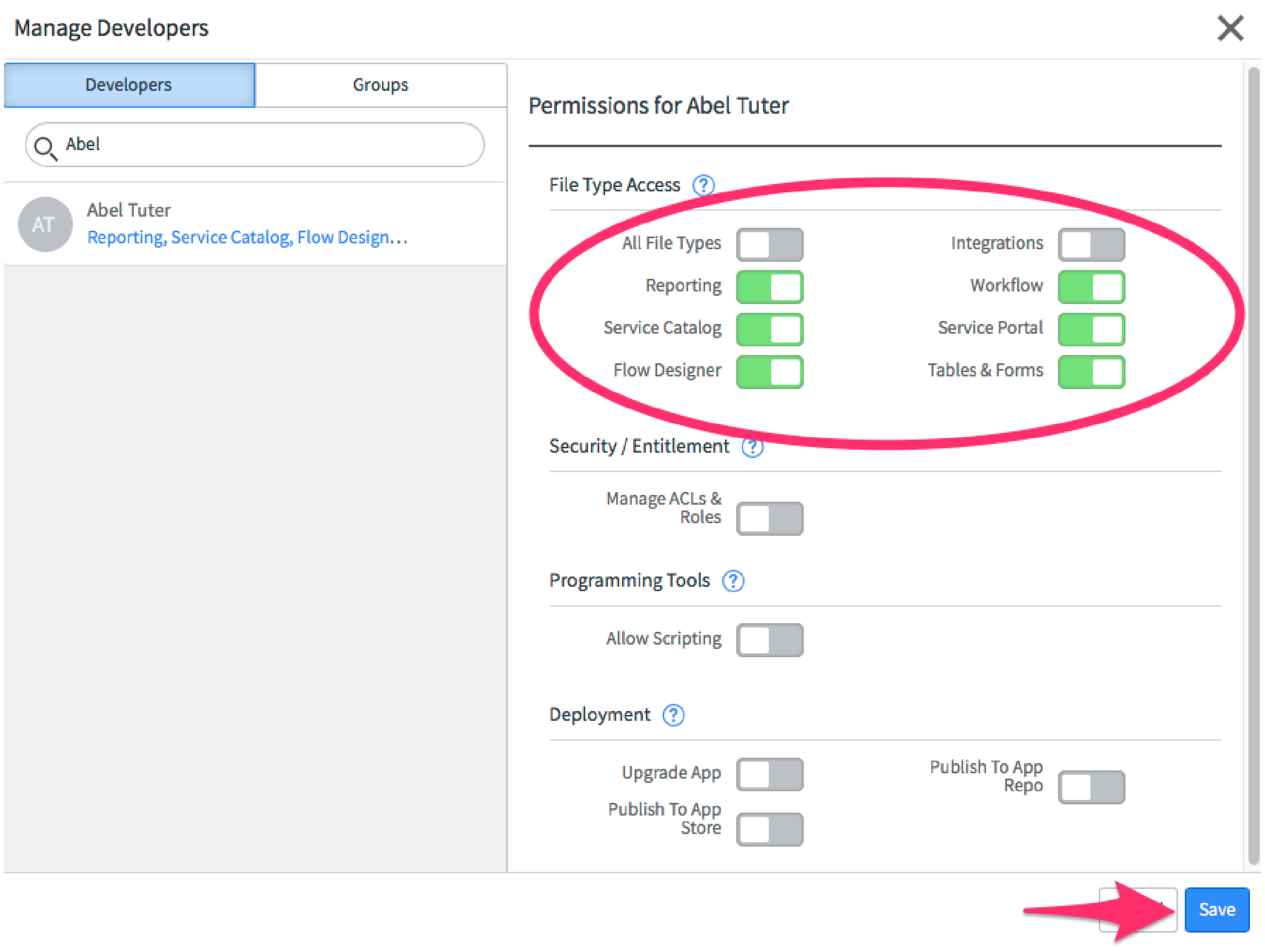
Enable Session Debug

Click Developer or Group, and start typing the individual developer's or developer group's name into the input box, then select the developer or group you are adding:



Add or remove developers and manage their permissions to platform provided resources.

Finally, set the desired permissions and save the record. In this example, we've chosen a limited set of permissions appropriate for a **Citizen Developer** role:



Manage Developers

Developers | Groups

Abel

Abel Tuter
Reporting, Service Catalog, Flow Design…

Permissions for Abel Tuter

File Type Access

| | | | |
|---|---|---|---|
| All File Types | ☐ | Integrations | ☐ |
| Reporting | ☑ | Workflow | ☑ |
| Service Catalog | ☑ | Service Portal | ☑ |
| Flow Designer | ☑ | Tables & Forms | ☑ |

Security / Entitlement

| | |
|---|---|
| Manage ACLs & Roles | ☐ |

Programming Tools

| | |
|---|---|
| Allow Scripting | ☐ |

Deployment

| | | | |
|---|---|---|---|
| Upgrade App | ☐ | Publish To App Repo | ☐ |
| Publish To App Store | ☐ | | |

This developer will now be able to work on our application, within the limitations set above.

Other roles would require different settings. For a full listing of each option and how it affects permissions, please see the developer & deployment permissions listing in the ServiceNow Docs site.

**Documentation**: Delegated development and deployment