ClickDeploy.io

Enterprise Release Management Implementation Guide

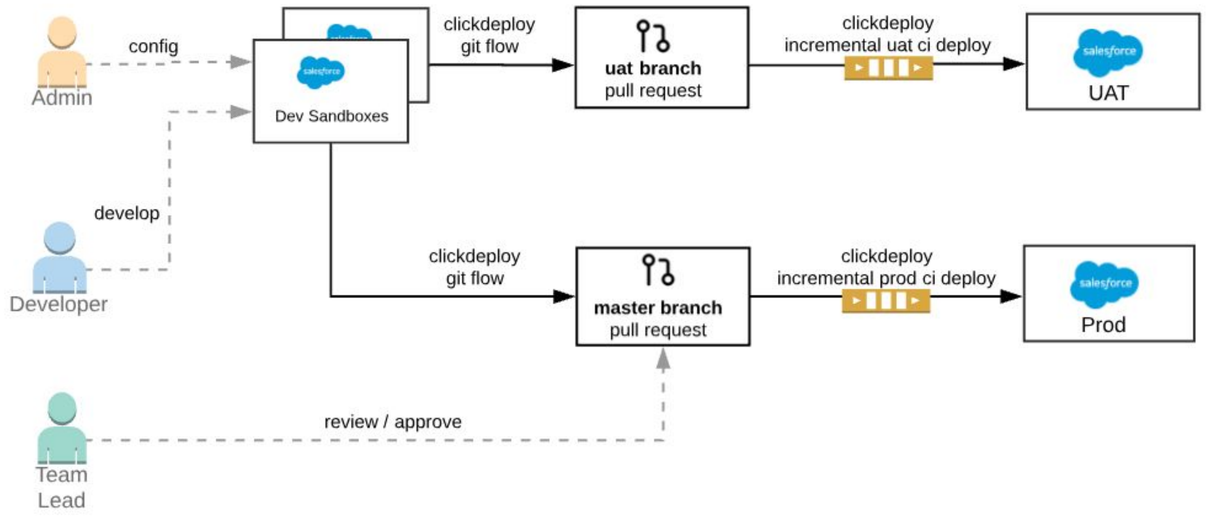
Overview

ClickDeploy Enterprise release process is carefully crafted to address a number of key challenges faced by many Salesforce teams around the world. By implementing ClickDeploy Enterprise, the following benefits are realized immediately.

- Complete version history of every single change.
- Prevent team members from overwriting each other.
- Ability to rollback a deployment.
- Streamline change review and approval process.
- Allow multiple changes to be merged, validated and ready for the release day.
- Support complex metadata types like Profile and Permission Set.
- Pre-production deployment report.
- Enable multiple project teams to work in parallel.
- Eliminate hours spent in manual coordination and troubleshooting of deployment errors.
- No more months long release. ClickDeploy enables features to be released daily or as soon as they are accepted in UAT.
- A simple and intuitive UX that empowers admins to work with source control seamlessly in a declarative fashion.
- A fully streamlined and automated release process that allows your team to focus on delivering business value faster.

This document describes the process flow and a step-by-step guide on how to set this up in your production environment.

The Process



Key steps

- Admins/devs build and test features in their dev sandboxes.
- The admin/dev then submits a pull request against *uat* branch via ClickDeploy git flow. Once the pull request is approved and merged, ClickDeploy CI job picks up the changes and kick off the incremental CI build to deploy the feature to the UAT environment.
- Once the feature is tested and accepted in UAT, the admin/dev submits a second pull request targeting the *master* branch.
- Notifications are sent to the team leads for approval. Upon approval, the changes are queued up in *master* ready for production deployments. Production deployments can be triggered automatically via webhook, time schedule or manual.

Notes

- Two main branches *master* and *uat*. Any commits to *master* branch get deployed to production org. Any commits to *uat* branch get deployed to UAT org.
- The CI jobs is setup as incremental builds which deploy only changes since the last successful build rather than deploying the entire branch. This helps to speed up deployments as well as minimize the risk of overwriting unrelated changes in production.

Implementation

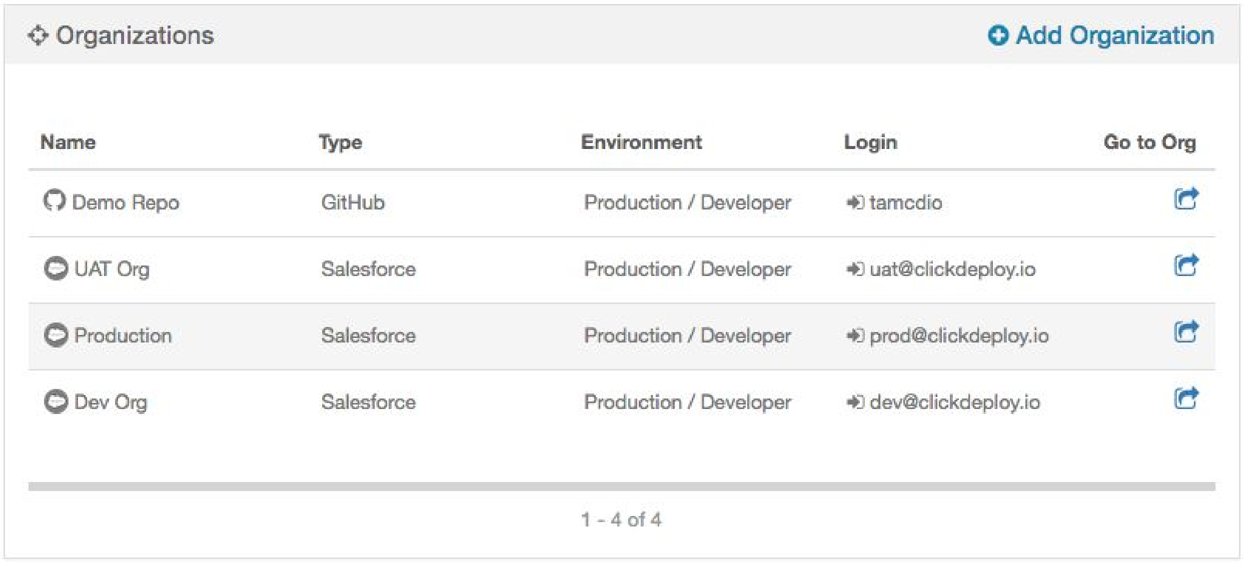
The process above takes approximately 30 minutes to implement using ClickDeploy Enterprise.

What you need

- A git repository. ClickDeploy supports Bitbucket, Github, Gitlab and Azure DevOps.
- A ClickDeploy Enterprise (or trial) account.

Step 1 - Connecting orgs

Sign in to ClickDeploy and add your Salesforce orgs and Git repository. On completion, the org list should look like below:



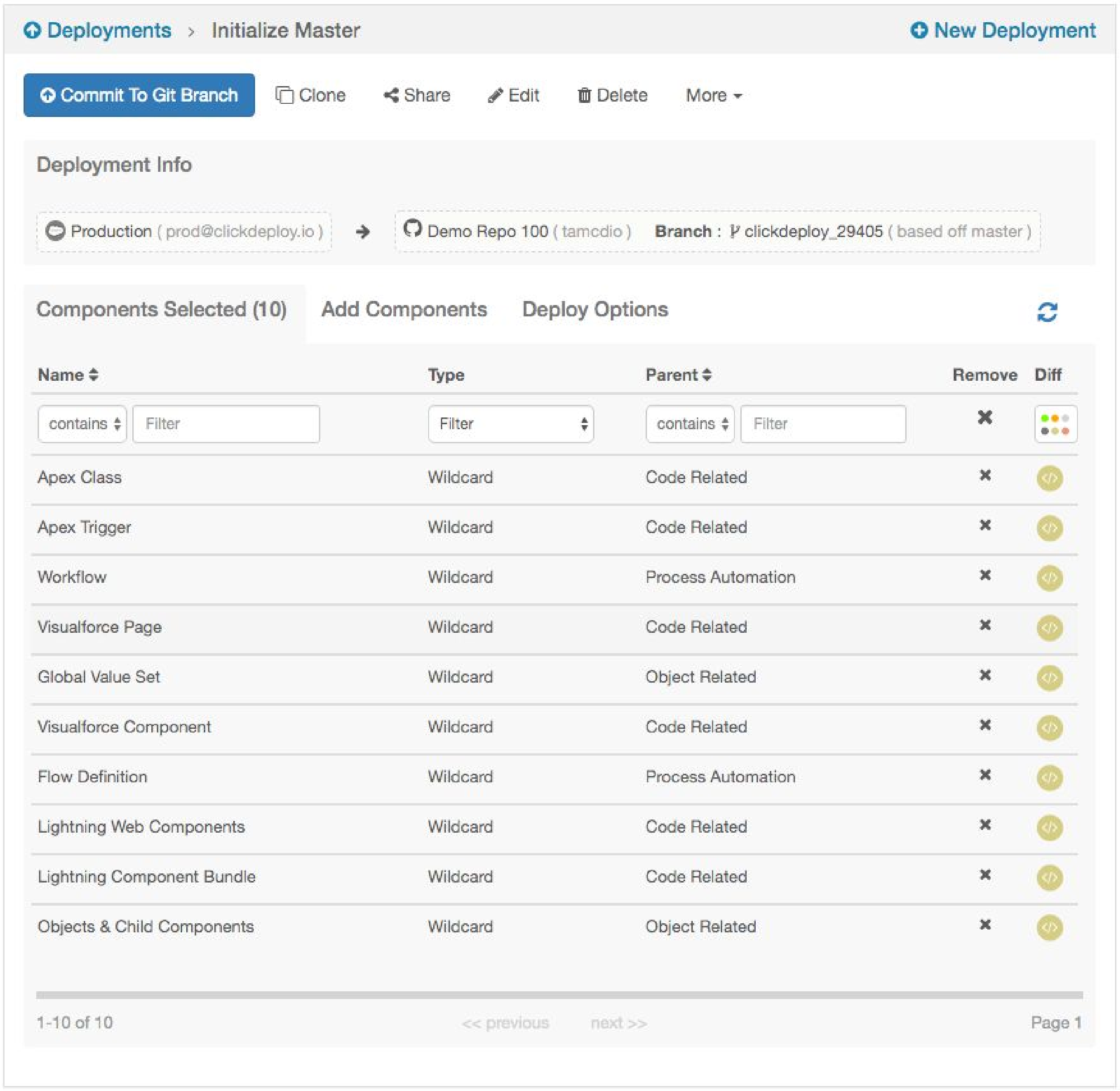
Name	Type	Environment	Login	Go to Org
Demo Repo	GitHub	Production / Developer	tamcdio	
UAT Org	Salesforce	Production / Developer	uat@clickdeploy.io	
Production	Salesforce	Production / Developer	prod@clickdeploy.io	
Dev Org	Salesforce	Production / Developer	dev@clickdeploy.io	

1 - 4 of 4

Step 2 - Initializing git repository

At the starting point, your *master* branch and your *uat* branch should contain metadata that reflects your production and uat org, respectively.

To initialize master branch with production metadata, create a deployment from your *Production* org to your git repo targeting *master* branch, like below.



Deployments › Initialize Master — New Deployment

Commit To Git Branch | Clone | Share | Edit | Delete | More

Deployment Info

Production (prod@clickdeploy.io) → Demo Repo 100 (tamcdio) Branch : clickdeploy_29405 (based off master)

Components Selected (10) | Add Components | Deploy Options

Name	Type	Parent	Remove	Diff
Apex Class	Wildcard	Code Related	✖	
Apex Trigger	Wildcard	Code Related	✖	
Workflow	Wildcard	Process Automation	✖	
Visualforce Page	Wildcard	Code Related	✖	
Global Value Set	Wildcard	Object Related	✖	
Visualforce Component	Wildcard	Code Related	✖	
Flow Definition	Wildcard	Process Automation	✖	
Lightning Web Components	Wildcard	Code Related	✖	
Lightning Component Bundle	Wildcard	Code Related	✖	
Objects & Child Components	Wildcard	Object Related	✖	

1-10 of 10 << previous next >> Page 1

Wildcard allows us to select all components of a certain type without having to name each component explicitly. You can add wildcards via *Add Components > Type > Wildcard*. We recommend to start with *Code Related*, *Object Related* and *Process Automation* group. You can add more metadata types at a later stage. Do not add Profiles at this stage, see [Working with Profiles section]() to learn more.

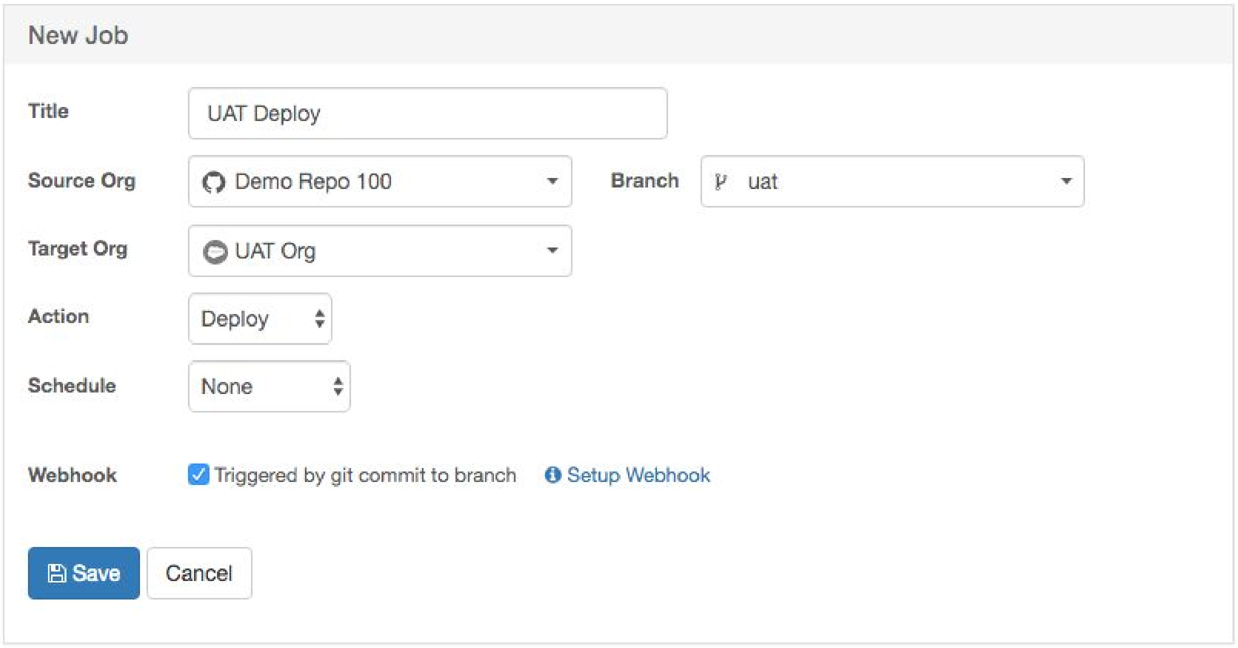
Under *Deploy Options*, set *Source Format* to *Salesforce DX*. See [Source Format Selection]() to understand the reasons.

Hit *Commit To Git Branch* and merge the changes to master.

Now that your master branch reflects production. Create a *uat* branch based off *master* branch and follow the same step above. This time, change source to UAT org and target to *uat* branch.

Step 3 - Setting up CI jobs

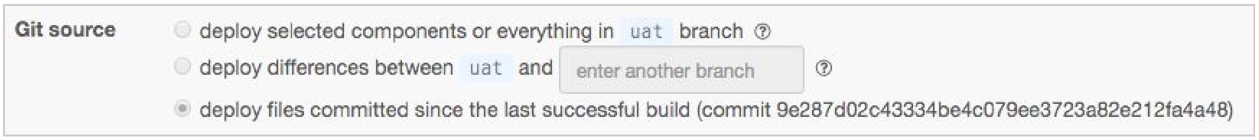
Create a new CI job that deploys from *uat* branch to UAT org, like below.



New Job

Title	UAT Deploy
Source Org	Demo Repo 100
Branch	uat
Target Org	UAT Org
Action	Deploy
Schedule	None
Webhook	✓ Triggered by git commit to branch

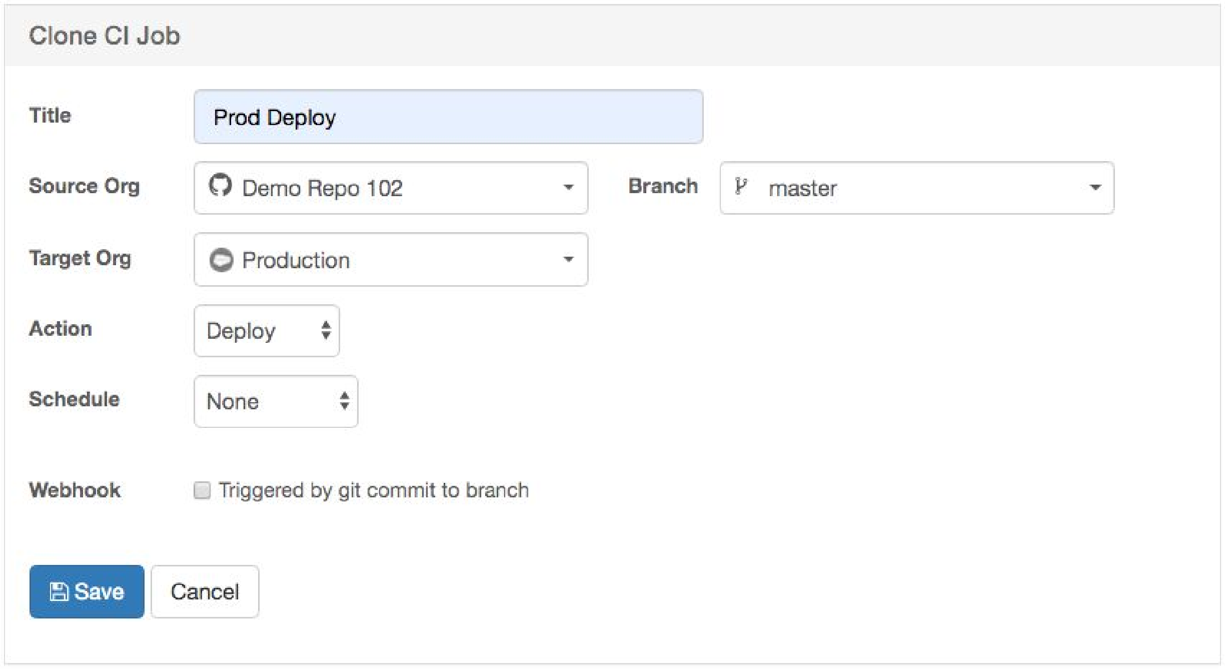
After save, edit *Deploy Options* and change *Git Source* option to *deploy files committed since the last successful build*, like below.



Git source
- deploy selected components or everything in uat branch
- deploy differences between uat and enter another branch
- [✓ deploy files committed since the last successful build (commit 9e287d02c43334be4c079ee3723a82e212fa4a48)]

By default, the last successful build marker points to the head of the branch. If you click on *Review Pending Changes*, the queue should contain zero components at this point.

Now that your UAT job is all set, you can create the Production Deploy job by cloning the UAT job and change branch to *master* and target to *Production* org, like below:

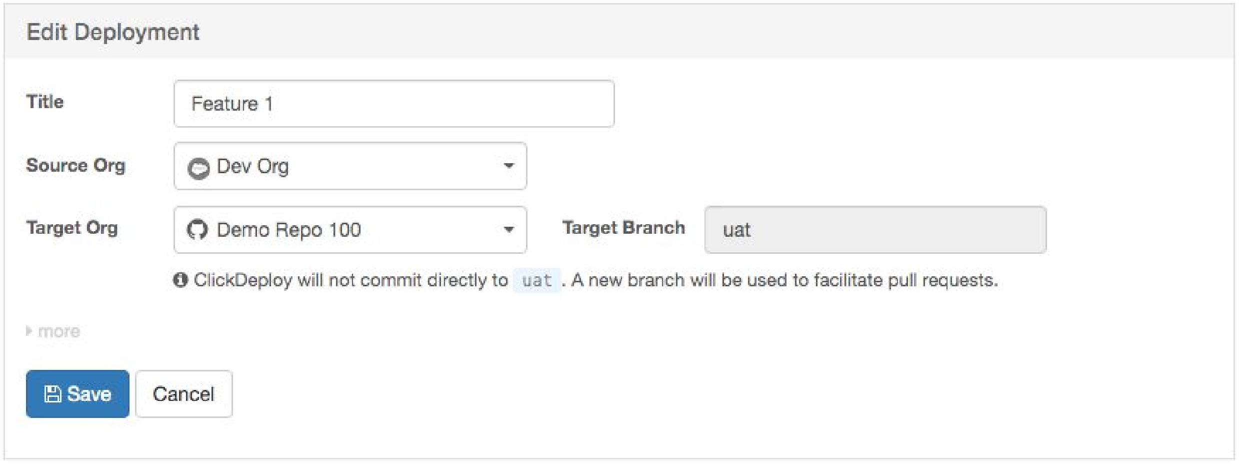

Clone CI Job

Title	Prod Deploy
Source Org	Demo Repo 102
Branch	master
Target Org	Production
Action	Deploy
Schedule	None
Webhook	☐ Triggered by git commit to branch

Remember to uncheck the webhook trigger to keep production deployment manual to start with. You can enable *schedule* option as you feel more comfortable with the process later on.

Step 4 - Submitting your first change request

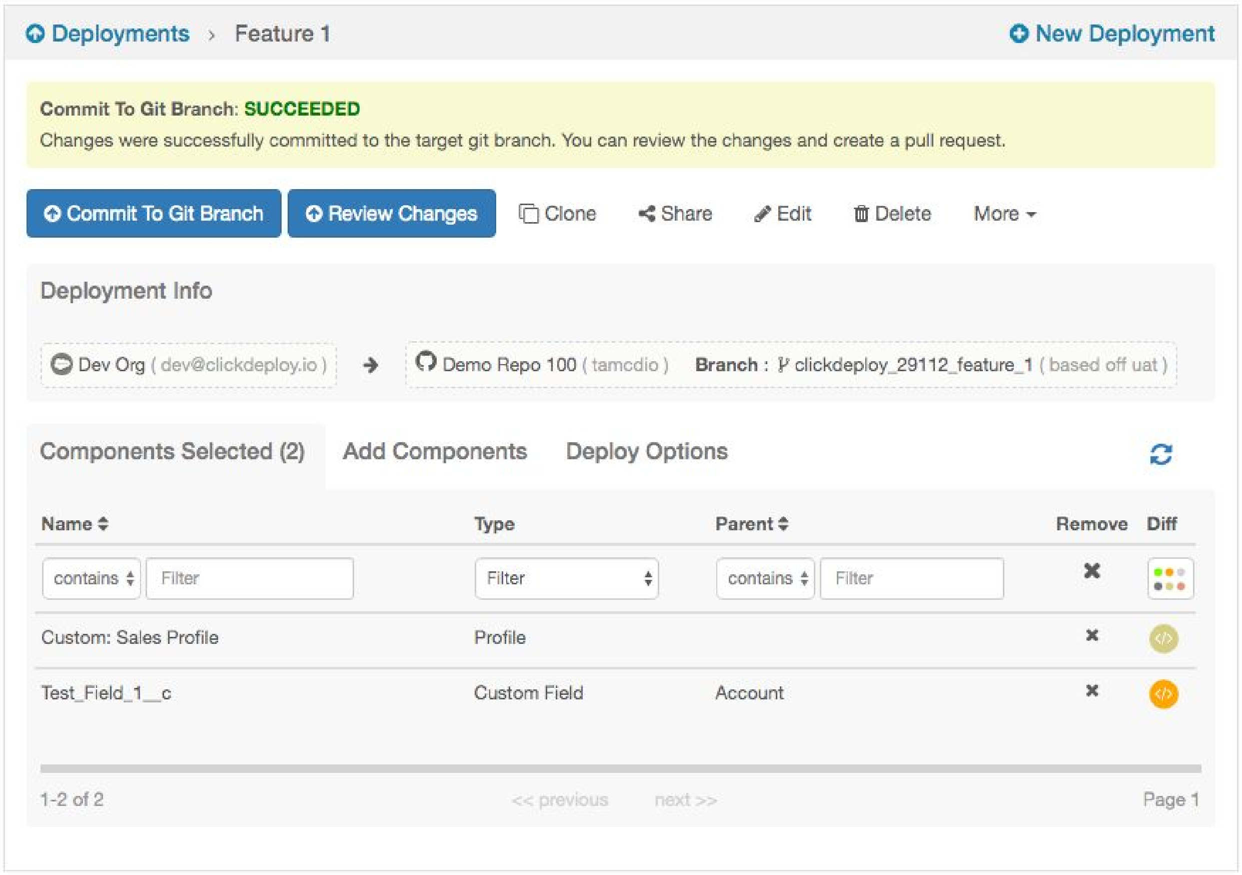
Create a new deployment from your dev sandbox to git targeting the *uat* branch, like below:


Edit Deployment

Title	Feature 1
Source Org	Dev Org
Target Org	Demo Repo 100
Target Branch	uat

ClickDeploy will not commit directly to `uat`. A new branch will be used to facilitate pull requests.

After save, add the changed components and click ‘Commit to Git Branch’.



Commit To Git Branch: SUCCEEDED
Changes were successfully committed to the target git branch. You can review the changes and create a pull request.

Deployment Info

Dev Org (dev@clickdeploy.io) → Demo Repo 100 (tamcdio) **Branch :** clickdeploy_29112_feature_1 (based off uat)

Name	Type	Parent	Remove	Diff
Custom: Sales Profile	Profile		✖	
Test_Field_1__c	Custom Field	Account	✖	

1-2 of 2 << previous next >> Page 1

Click *Review Changes* and submit the pull request. After the pull request is merged, the change should be deployed to UAT automatically.

Once the change is tested in UAT, clone the deployment and follow the same step above. This time, change the target branch to *master* in order to have the feature released to production.

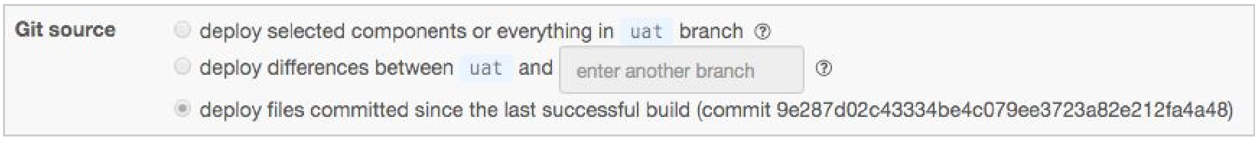
Review your Production CI job, click on *Review Pending Changes*, this should show you what is currently in the queue and the differences against production.

Hit *Run Now* and the feature should be deployed to Prod environment.

Fine-tuning

Understand incremental diff deployment

Incremental build contains only files committed since the last successful build which is indicated by a build marker. Every time a deployment is successful, the build marker is automatically moved to the head of the branch.



Git source

- deploy selected components or everything in `uat` branch
- deploy differences between `uat` and enter another branch
- [✓ deploy files committed since the last successful build (commit 9e287d02c43334be4c079ee3723a82e212fa4a48)]

This means the *pending change queue* is reset after every successful deployment.

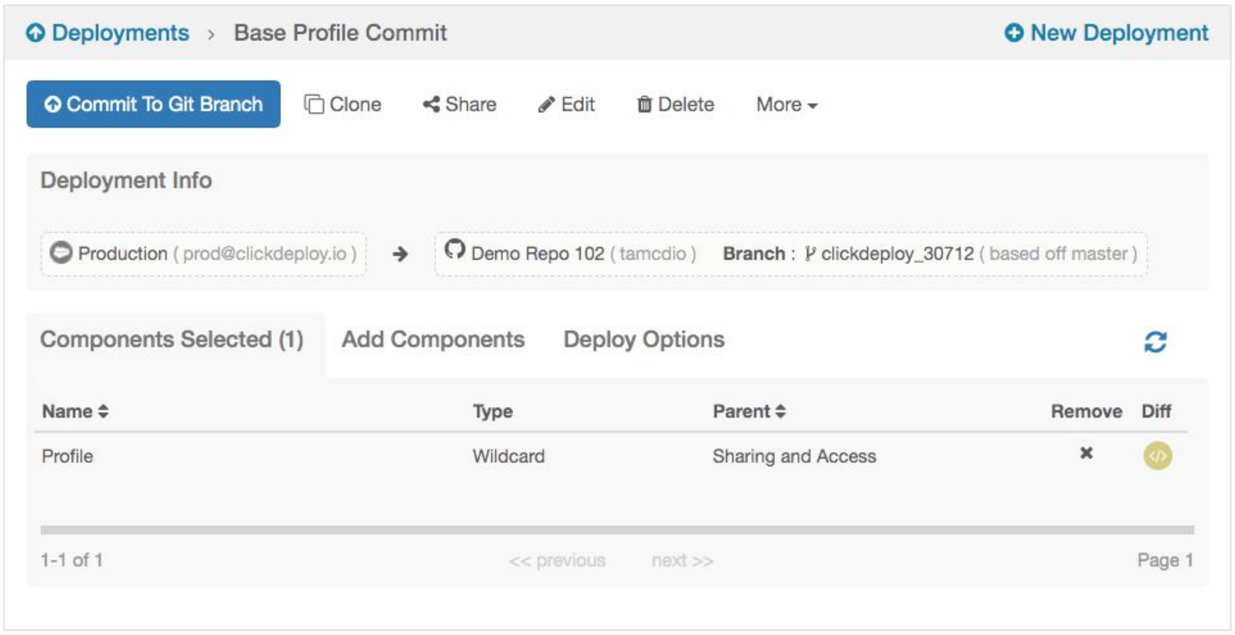
No manual intervention is required in general. However, if you ever need to change the build marker to a specific point, simply edit *Deploy Options* and point to the new commit id.

Note that for an incremental validate CI job, a successful validation does not move the build marker automatically. The build marker for a validate CI job is only moved when the equivalent incremental deploy CI job build is deployed successfully. This is by-design to ensure that the incremental validate build always capture the same changes as the incremental deploy build.

Working with profiles

A full Profile file can contain 50k lines of xml which makes it unfit for the purpose of source control driven deployments and merge resolution. In order to work with profiles effectively, commit only the base profiles when you initialize your git repository for the first time.

You can commit only the base profiles by creating a separate commit deployment from your Prod org to your git repo and include only the Profile wildcard (excluding any other wildcards), like below. This way, the profiles will not carry any other related permissions.


Deployments › Base Profile Commit

New Deployment

Commit To Git Branch | Clone | Share | Edit | Delete | More

Deployment Info

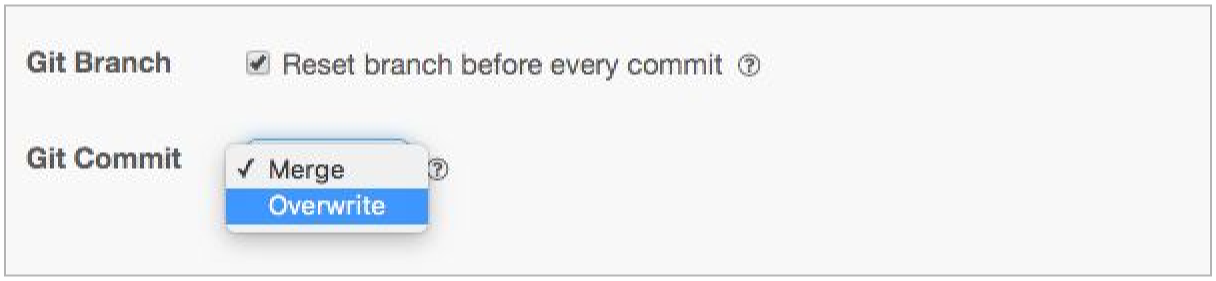
Production (prod@clickdeploy.io) → Demo Repo 102 (tamcdio) **Branch :** clickdeploy_30712 (based off master)

Components Selected (1) | Add Components | Deploy Options

Name	Type	Parent	Remove	Diff
Profile	Wildcard	Sharing and Access	✖	●

1-1 of 1 << previous next >> Page 1

After a few months of development, your profile files will eventually grow to a certain size. At that point, you can decide to trim the profiles and bring it back to its starting baseline by repeating the above commit deployment, this time, edit *Deploy Options* and change the *Git Commit* option to *Overwrite*.


Git Branch ✓ Reset branch before every commit

Git Commit ✓ Merge / Overwrite

Overwrite means that ClickDeploy will simply take the base profile files and overwrite the large profile files in your Git repo.

Source format selection

We recommend to use Salesforce DX source format instead of the old Mdapi source format for the following reasons:

- Incremental diff deployment works better with DX format. When you commit a single field, the diff would include only this single field in the incremental build. In the case of Mdapi

format, since the field is part of a big object file, the entire object is included which contains a lot more than just the single changed field.
- The breakdown DX code format makes merging easier.
- DX source format is a prerequisite for developing Lightning Web Components.

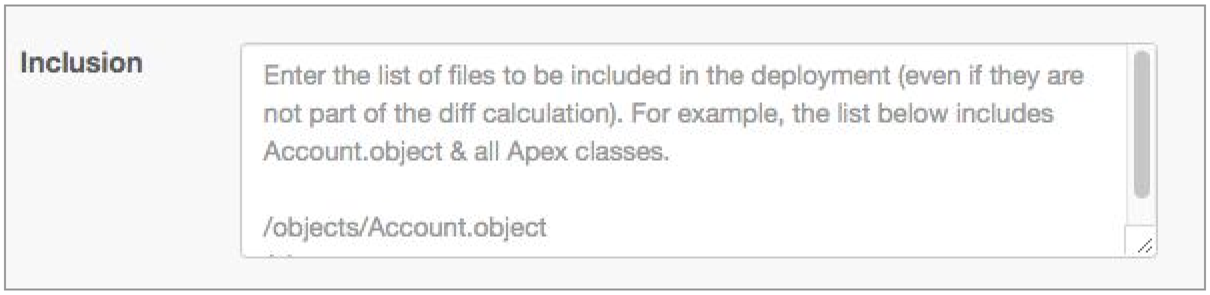
Note that DX is a spectrum of technologies including a new code format, CLI, scratch org, source tracking, and packaging 2.0. At this stage, **the only part of DX that we recommend teams to leverage is the new code format**. The remaining parts still need a lot more time to reach production maturity. Scratch org and packaging 2.0 requires a significant amount of upfront investment in breaking down the org into modules and the overhead of managing dependencies between these modules cannot be understated.

Switching from Mdapi format to DX format does **not** change your development workflow. You can continue to develop against sandboxes (instead of scratch orgs) or use your favourite developer tools (VS Code, Illuminated Cloud, Welkin Suite). All these tools now support using DX format against sandboxes.

This does not mean that you cannot use Mdapi format. ClickDeploy supports both DX and Mdapi source format. Just keep in mind that any deployment strategy that involves calculating differences would normally work more effectively when the source is broken down into smaller files.

Understand inclusion option

Incremental deployment strategy is a great way to speed up your CI builds. Occasionally, there could be specific scenario where you might want to include certain files in your CI builds even if they have not been changed. This can be achieved by leveraging ClickDeploy *inclusion option*.


Inclusion

Enter the list of files to be included in the deployment (even if they are not part of the diff calculation). For example, the list below includes Account.object & all Apex classes.

/objects/Account.object

Simply edit *Deploy Options*, and enter paths of the files you wish to include in the CI build and these files will be included together with the result of the diff calculation.