“Clicks, Not Code” to “Clicks Are Code”:

Add Git to your #AwesomeAdmin Superpowers

David Reed | Andrés Catalán

February 5, 2019

Andrés Catalán

Data Scientist and Consultant, Slalom

10x certified

@sfdx__andres

David Reed

Salesforce Application Architect, Radian

8x certified

@aoristdual

Agenda

1. Source Code and Salesforce DX for Admins

   Metadata and source code - moving metadata - saving history - building in text form

2. Using Git to Manage Salesforce Source Code

   Version control - “source of truth” - repositories - checking in - pushing and pulling - merging and branches

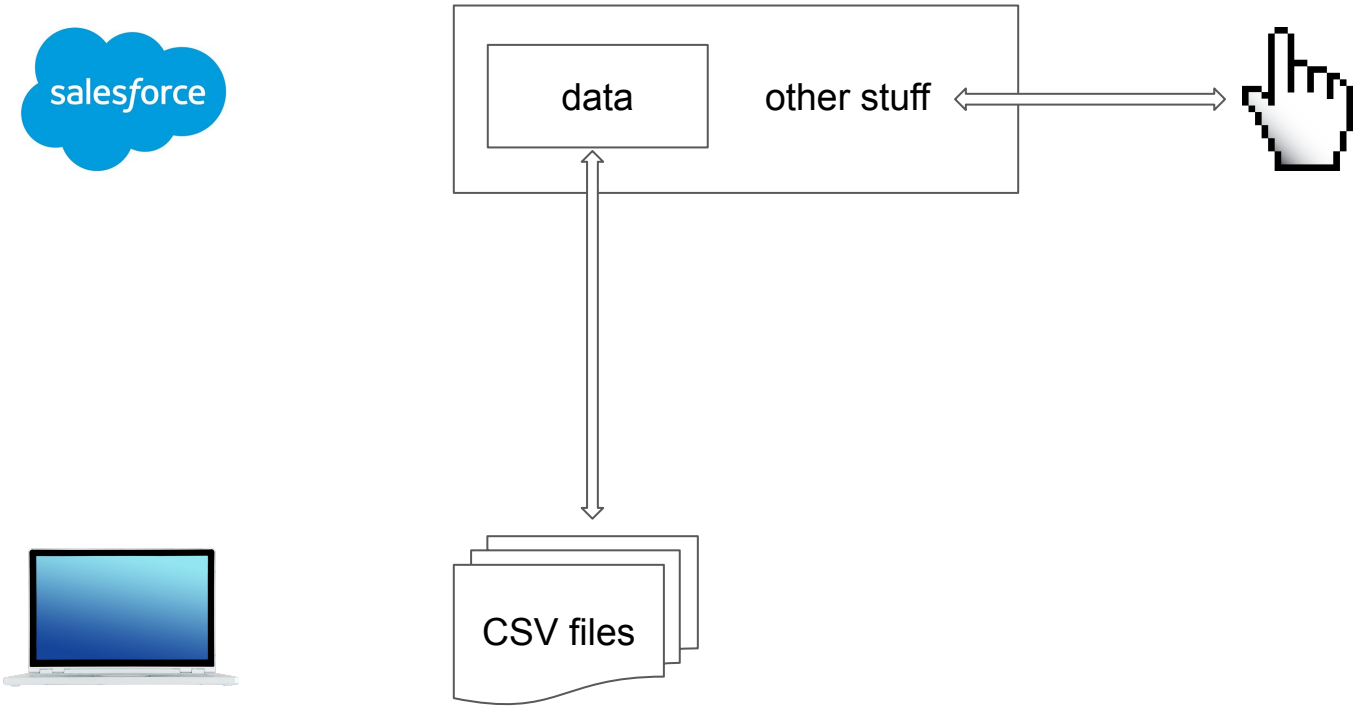
3. Hands-on Workshop

   Build an application with everyone else in this room - at the same time.

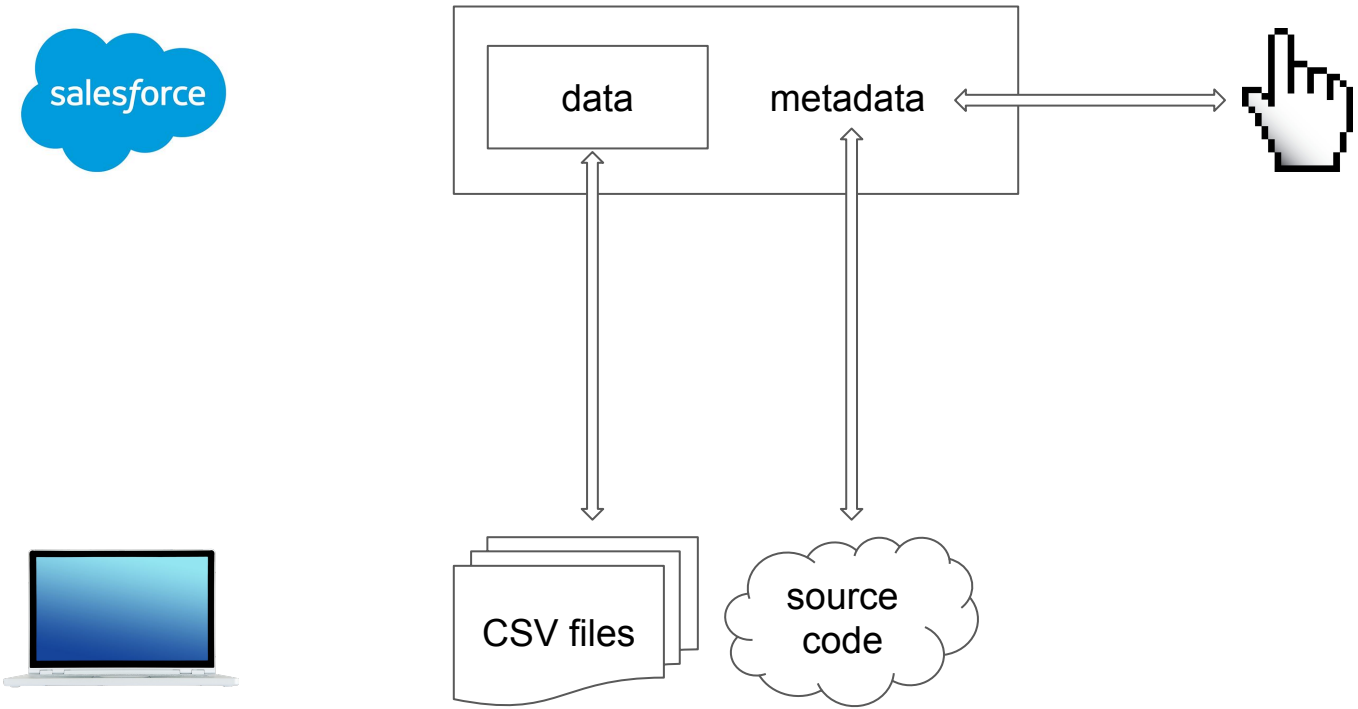
While we talk...

- Haven't done the pre-work for this workshop yet? Now's a good time!
  - All these steps are in the Meetup event and our Git repo.
  - Sign up for a Salesforce Developer Edition: https://developer.salesforce.com/signup
  - Turn on Dev Hub: https://developer.salesforce.com/docs/atlas.en-us.sfdx_setup.meta/sfdx_setup/sfdx_setup_enable_devhub.htm
  - Sign up for a free GitHub account: https://github.com/join
  - Install the Salesforce DX tools: https://developer.salesforce.com/docs/atlas.en-us.sfdx_setup.meta/sfdx_setup/sfdx_setup_install_cli.htm#sfdx_setup_install_cli
  - Install Visual Studio Code (https://code.visualstudio.com/)
  - Install the Salesforce DX extension pack in Visual Studio Code: https://marketplace.visualstudio.com/items?itemName=salesforce.salesforcedx-vscode

“Clicks, Not Code”



Clicks Are Code!



Source Code

(Almost) Everything Is Source Code

- Source code is an external representation of the metadata in an org.
- Source code can be used to move metadata between orgs.
- Source code can be manipulated and used to create metadata in an org.

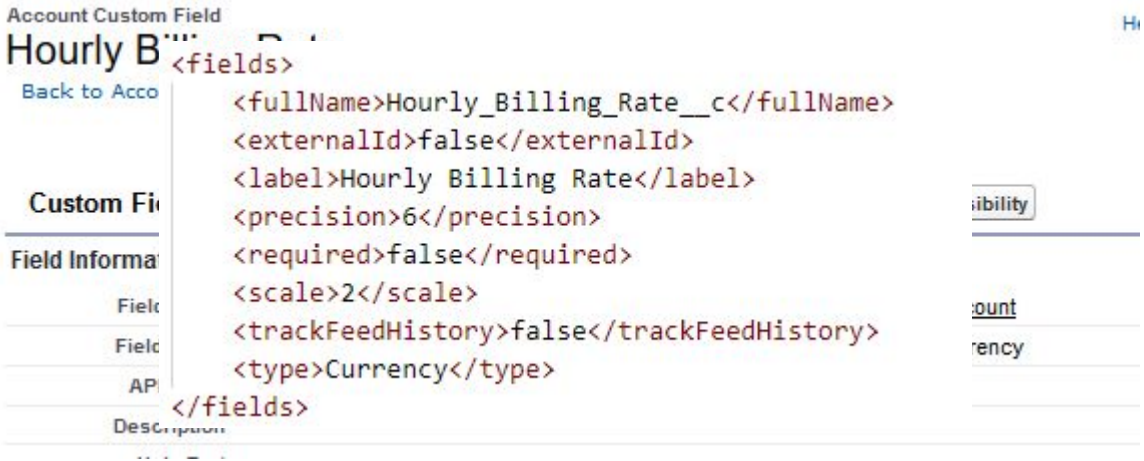
- **More of your Salesforce org is source code than you think!**
  - **See Salesforce Metadata Coverage Report**

(Almost) Everything Is Source Code

(Almost) Everything Is Source Code

```
<?xml version="1.0" encoding="UTF-8"?>
<Flow xmlns="http://soap.sforce.com/2006/04/metadata">
    <assignments>
        <name>myVariable_waitStartTimeAssignment</name>
        <label>myVariable_waitStartTimeAssignment</label>
        <locationX>0</locationX>
        <locationY>0</locationY>
        <assignmentItems>
            <assignToReference>myVariable_waitStartTimeVar
            <operator>Assign</operator>
            <value>
                <elementReference>$Flow.CurrentDateTime</e
            </value>
        </assignmentItems>
        <connector>
            <targetReference>myDecision</targetReference>
        </connector>
    </assignments>
    <decisions>
        <processMetadataValues>
            <name>index</name>
```

(Almost) Everything Is Source Code



✓

(Almost) Everything Is Source Code


```
<?xml version="1.0" encoding="UTF-8"?>
<Layout xmlns="http://soap.sforce.com/2006/04/metadata">
    <excludeButtons>ChangeRecordType</excludeButtons>
    <excludeButtons>Submit</excludeButtons>
    <layoutSections>
        <customLabel>false</customLabel>
        <detailHeading>false</detailHeading>
        <editHeading>true</editHeading>
        <label>Account Information</label>
        <layoutColumns>
            <layoutItems>
                <behavior>Required</behavior>
                <field>Name</field>
            </layoutItems>
            <layoutItems>
                <behavior>Edit</behavior>
                <field>ParentId</field>
            </layoutItems>
            <layoutItems>
                <behavior>Edit</behavior>
                <field>Hourly_Billing_Rate__c</field>
            </layoutItems>
        </layoutColumns>
```

Source Used to Be **Hard** to Manage

1. Build your toolset
    a. Download ANT (open source scripting language)
    b. Download a Jar file (API version-specific!) from Salesforce to tell ANT how to talk to your org
    c. Store all your credentials, including access token, in a local plain text file (super safe …not!)
    d. Good luck!
2. Put together a manifest of metadata to get/send (package.xml)
    a. Learn and understand the metadata model (very idiosyncratic!)
3. Rinse and repeat (and repeat, and repeat) to get stuff done
    a. Huge, monolithic metadata files
    b. Want to delete something? Fun!
    c. When you send metadata to your org, it has to be internally and externally consistent!

The **Salesforce Command Line Interface (CLI)** solves #1.
Don't want to deal with the command line? **Salesforce's official VSCode plugins** "solve" Salesforce CLI.

The **DevHub** (eventually) solves #2 and #3.

Demo:
Classic Metadata Wrangling

Salesforce CLI: The Ohana's New Best Friend (part 1)

- Salesforce CLI (or **sfdx**) gives you new superpowers
  - It has **aliases** to hold "the keys to the kingdom" for all your orgs.
    - Log in ***once*** using OAuth — even if you change passwords
    - Go to the Command Palette or command line when you want to log into your org
    - Type the command... Boom! Open! No password!

```
sfdx force:auth:web:login -a <name_of_alias>
> SFDX: Authorize an Org
sfdx force:org:open -u <name_of_alias>
> SFDX: Open Default Org
```

  - It can be used to replace ANT's most basic use cases, *if you have to*:

```
sfdx force:mdapi:retrieve -u <name_of_alias> -k <package_file> -r <folder>
sfdx force:mdapi:deploy -u <name_of_alias> -d <folder> -w <num_minutes>
```

Do try this at home!

Scratch Orgs: Welcome to the Future

The "Source of Truth" Problem: *Where* is the "true state" of your system?

Scratch Orgs: Welcome to the Future

Infrastructure as Code: Disposable orgs!

Demo:
Developing in Scratch Orgs

Salesforce CLI: The Ohana's New Best Friend (part 2)

- Salesforce CLI (or **sfdx**) gives you new superpowers
  - It creates and maintains projects with metadata source code to be sent to scratch orgs:

```
sfdx force:project:create -n <project_name>
> SFDX: Create Project
```

  - It brings changes in the local folder to the scratch org (push) or vice versa (pull)

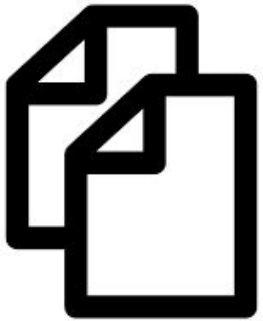
```
sfdx force:source:pull -u <name_of_alias>
> SFDX: Pull Source
sfdx force:source:push -u <name_of_alias>
> SFDX: Push Source
```

How to Get and Use Salesforce Source Code

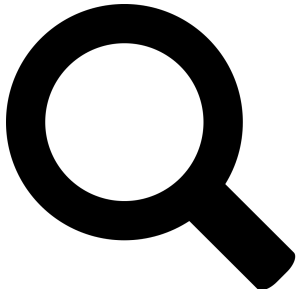
Three methods using the Salesforce CLI:

- ANT-like deployments using `mdapi`
- Source commands for sandboxes (beta)
- Scratch orgs and `source:[push|pull]`
    - This is what we're using today.

What Does Using Source Get Us?



Copy and paste metadata
- as source code



Global search and
replace, in your editor

Track changes with Git
(much more to come!)

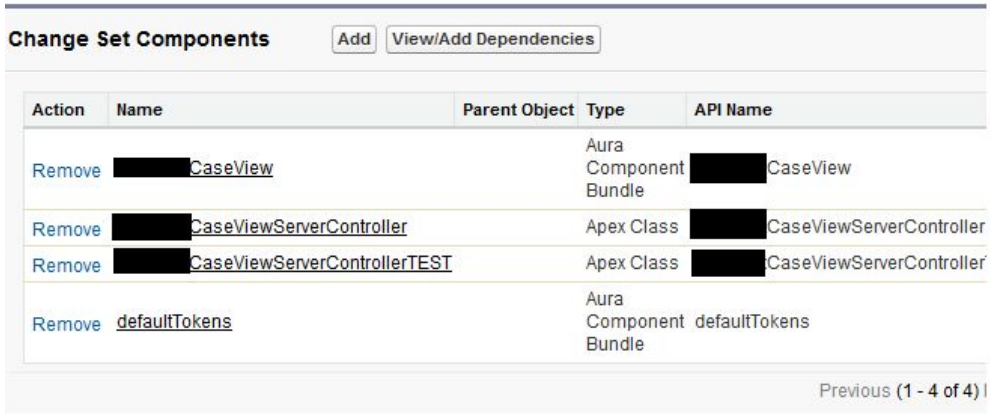
… plus full use of the latest development tools (SFDX) and interoperability with text-based tools.

Demo: Manipulating Metadata with Source Code

Version Control: Git

Tracking Changes At Scale

- Change sets represent a point in time.
  - No history tracking.
  - No support for team collaboration.
- Change sets are hard to manipulate and don't use source code.



Change Set Components Add | View/Add Dependencies

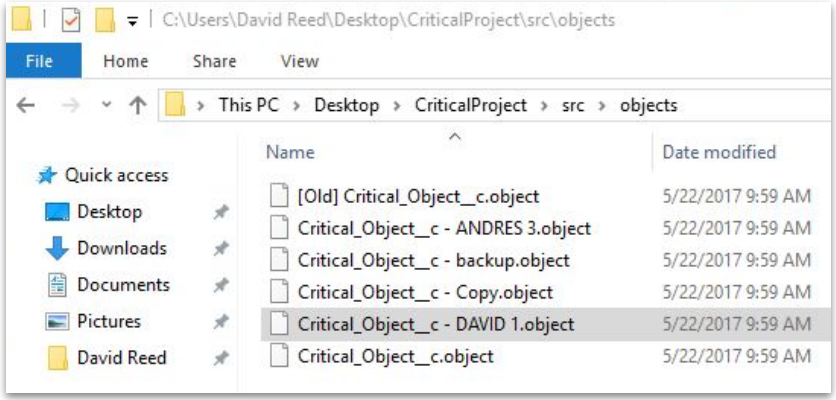
Action	Name	Parent Object	Type	API Name
Remove	[redacted]CaseView		Aura Component Bundle	[redacted]CaseView
Remove	[redacted]CaseViewServerController		Apex Class	[redacted]CaseViewServerController
Remove	[redacted]CaseViewServerControllerTEST		Apex Class	[redacted]CaseViewServerController
Remove	defaultTokens		Aura Component Bundle	defaultTokens

Previous (1 - 4 of 4)

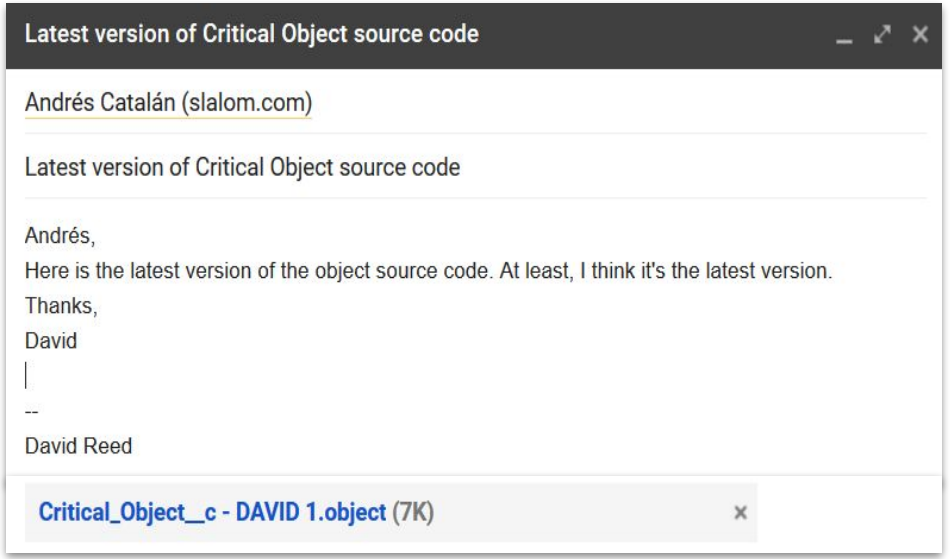
Tracking Changes At Scale

- Text source code is easy to manipulate, copy, track, and store.

  But how do we do this effectively at scale?



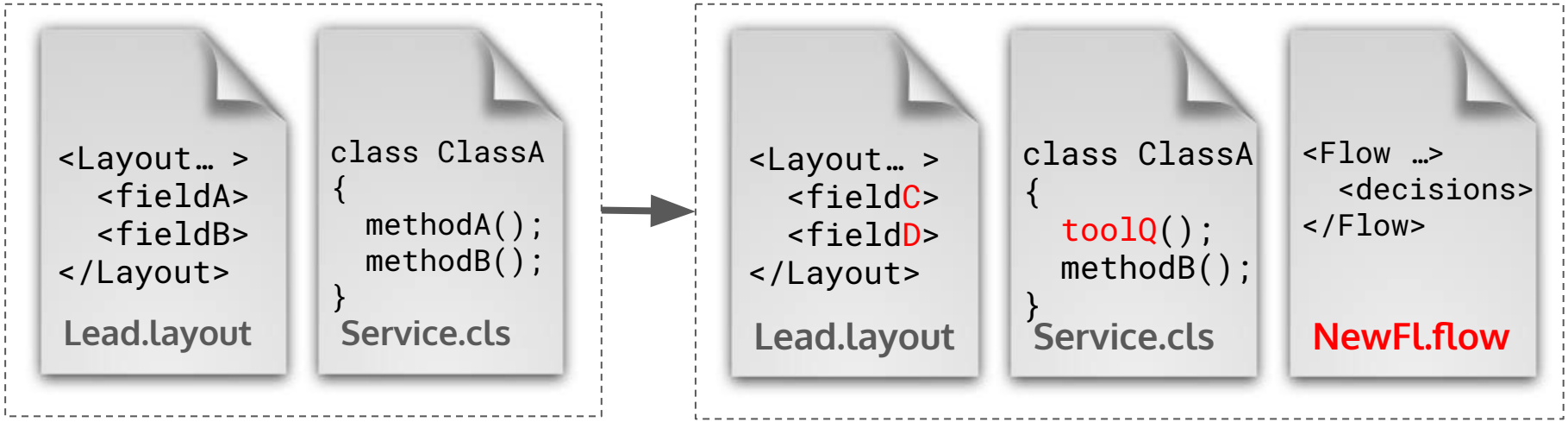
(Not like this).



Git: Scalable Version Control for One or Many Users

- Effectively track and share version history for files across a team - or by yourself.
- Visualize history and track the “what/when/who” of changes made.
- Bring back old versions you need today, and revert changes that cause problems.
- Handle conflicting changes and multiple parallel work streams (branches).

Commits: Core Unit of Change

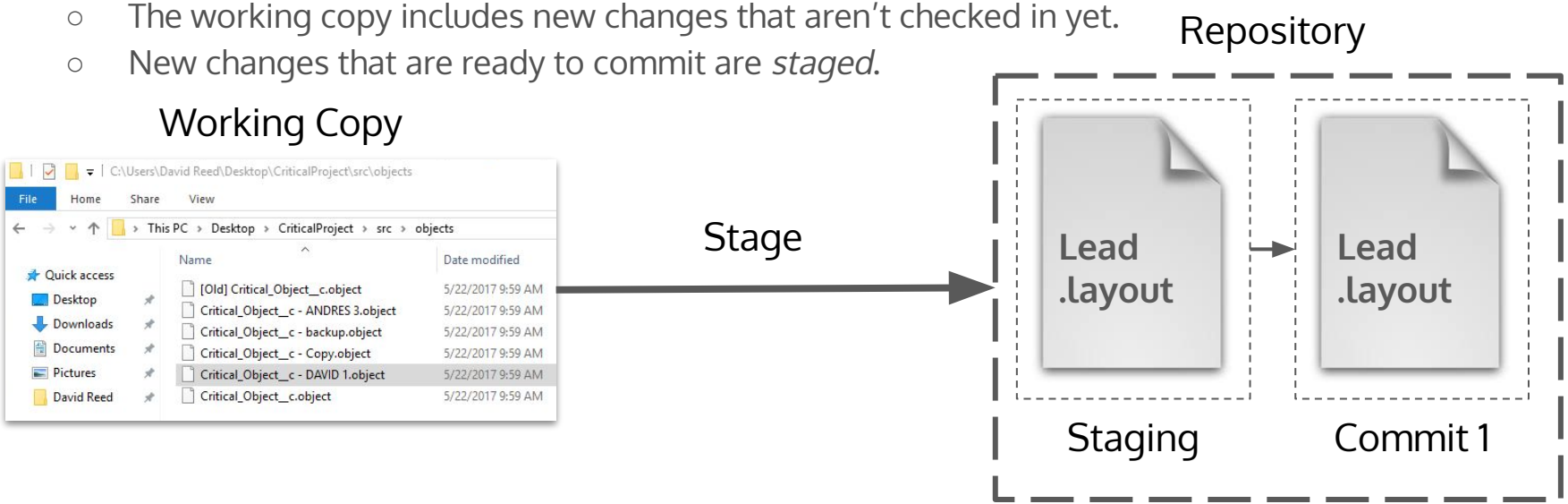


Commit 1

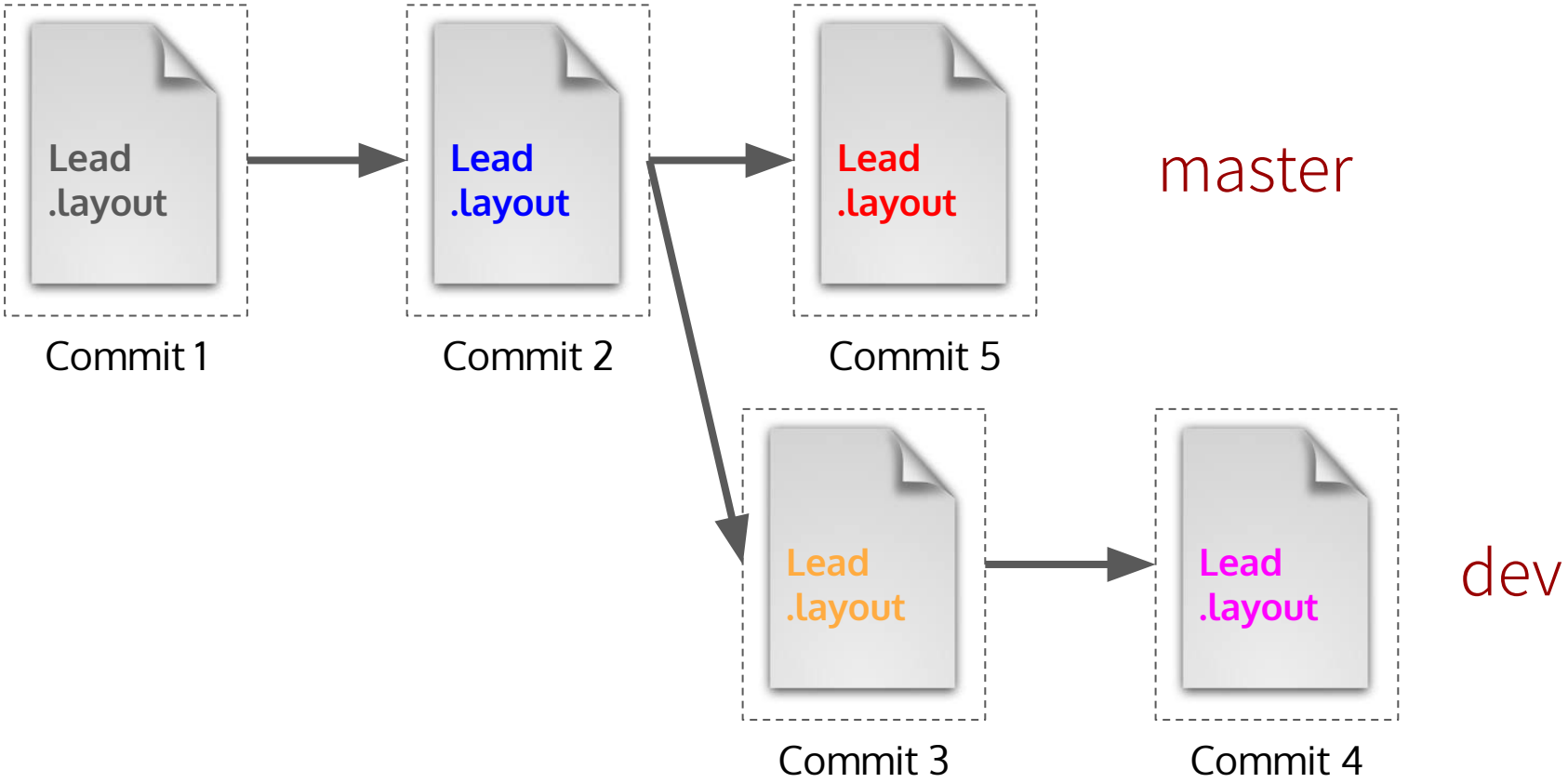
Commit 2

The Repository and the Working Copy

- The *repository* stores the whole history of your project.
- The *working copy* is the files you can see and work on.
  - The working copy includes new changes that aren't checked in yet.
  - New changes that are ready to commit are *staged*.

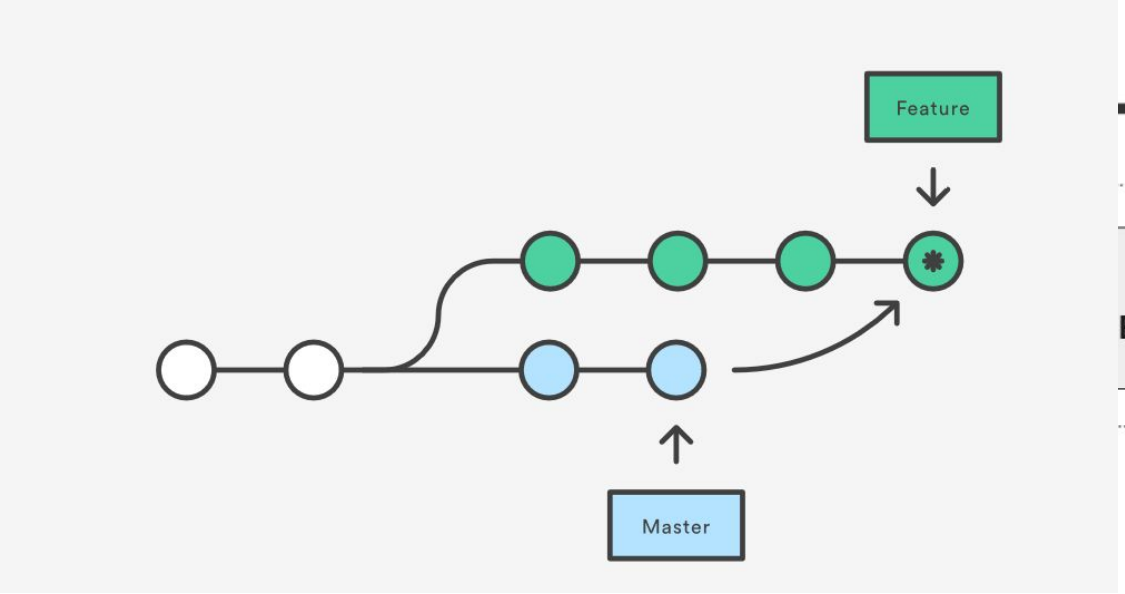


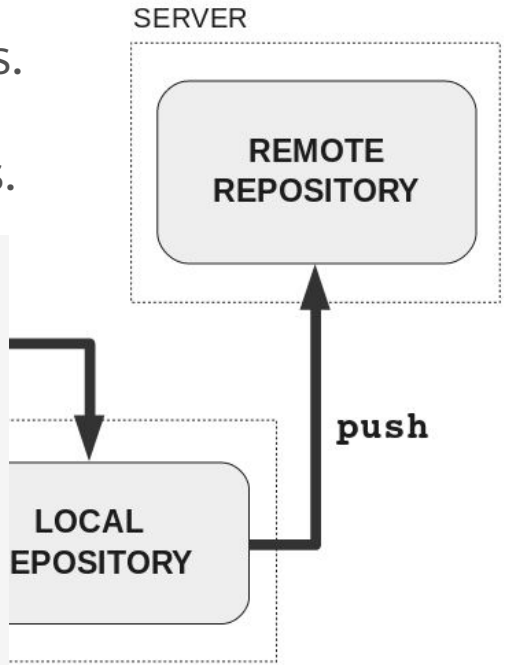
Branches: Tracking Sequences of Changes



Merging, Pushing, and Pulling

- *Merges* combine independent changes to the same files.
- *Push* and *pull* operations move commits between repos.





Demo: Sharing Work with Git

Hands-On Workshop

Steps

1. Fork Git Repository (https://github.com/davidmreed/clicks-are-code) — GitHub
2. Clone Git Repository to your Laptop — Visual Studio Code
3. Create a New Scratch Org
4. Push Metadata to Scratch Org
5. Launch Scratch Org
6. Make the Changes on your Card in Scratch Org — Salesforce
7. Pull Metadata to Working Copy
8. Review Changes
9. Add Changes to Git and Commit — Visual Studio Code
10. Push Changes to Repository
11. Create Pull Request in GitHub — GitHub

Resources

- Git Repository for Workshop: https://github.com/davidmreed/clicks-are-code
- Trailhead
  - Git and GitHub Basics: https://trailhead.salesforce.com/en/content/learn/modules/git-and-git-hub-basics
  - Quick Start: Salesforce DX: https://trailhead.salesforce.com/en/content/learn/projects/quick-start-salesforce-dx
  - Develop an App With Salesforce DX and Source Control: https://trailhead.salesforce.com/en/content/learn/projects/develop-app-with-salesforce-cli-and-source-control
  - App Development with Salesforce DX: https://trailhead.salesforce.com/en/content/learn/modules/sfdx_app_dev
- VS Code for Non-Developers by Jodie Miners
  - https://tddprojects.atlassian.net/wiki/spaces/SF/pages/595165185/VS+Code+for+Non+Developers
- Hosting Services: GitHub, GitLab, Bitbucket
  - Learn Git: https://www.atlassian.com/git/tutorials
  - GitHub Desktop (GUI): https://desktop.github.com/
- Advanced Reading: *Pro Git* by Scott Chacon (https://git-scm.com/book/en/v2)