**SFDX Basics**

Created By: Ryan Williams

**Trails:**

* [**https://trailhead.salesforce.com/en/content/learn/modules/sfdx\_app\_dev**](https://trailhead.salesforce.com/en/content/learn/modules/sfdx_app_dev)
  + Trailhead walkthrough of DX
  + Good Quick overview
* [**https://trailhead.salesforce.com/en/content/learn/trails/sfdx\_get\_started**](https://trailhead.salesforce.com/en/content/learn/trails/sfdx_get_started)
  + Has multiple modules about DX and other related topics
* [**Https://trailhead.salesforce.com/en/content/learn/modules/sfdx\_travis\_ci**](Https://trailhead.salesforce.com/en/content/learn/modules/sfdx_travis_ci)
  + Module about CI and CD with SFDX

**Notes:**

* Many of these commands can be replicated through using Visual Studio Code with the most recent version of the Salesforce CLI and Salesforce Certified Plugin.
* Add, -h or –help, you can display help for any command. Help provides information on what the command does, describes each parameter, and lists the short and long versions of the parameters.
  + sfdx force:auth:web:login -h
  + sfdx force:auth:web:login –help
* Adding an alias, -a parameter, to orgs/scratch orgs is highly encouraged. The alias can be used instead of the username of the org. This is beneficial because scratch orgs usernames can look like this, test-7emx29rtpx0y@example.com.
* As a best practice, immediately commit the source you brought into your project to a VCS. Scratch orgs are ephemeral and temporary, so you always want a backup of work you’ve saved locally.
  + 9
* You can create test data in one org and move it to multiple different orgs. This will cut down on time creating for each scratch org.
  + 11, 12
* You can create files from the command line as well. You need to specify the file type, name and location when creating the file.
  + 13, 14
* While validating your app you can absolutely use the same scratch org you’ve used during development to perform your testing, we recommend that you always start with a fresh scratch org. A fresh scratch org ensures you’ve properly externalized all your source from the org.
* You can access your instance url/custom domain from the command line. This is helpful when installing managed packages or getting access to things such as Dataloader.
  + 15, 16
* When retrieving or deploying code from/to orgs you need to retrieve, unzip, convert, push. This process allows for an easy migration between Scratch orgs.
  + 17, 18
* sfdx force:source:convert vs sfdx force:mdapi:convert
  + the difference is what is being converted. If you are converting from code you are working on you use source
  + if you are converting from code you just retrieved from an Org use mdapi
* To deploy code you have been working on to another Scratch Org you have to convert it to mdapi and then deploy it to the specific Org
  + 19, 20
* CI and CD are beneficial tools when paired with SFDX. The use of these tools allows, once set up, a user to push their work to a common Repo with will then deploy these changes to another location. These tools will also run through test scripts and will catch any errors before the deployment occurs. Catching bugs before they have a chance to move environments.

**Commands:**

1. sfdx force:auth:web:login -d -a DevHub

* Adding the -d flag sets this org as the default Dev Hub. Use the -a to set an alias for the org (something catchy like DevHub). An alias is much easier to remember than the unique Dev Hub username.

1. sfdx force:org:open -u DevHub

* Used to open the org that you want -u ‘Alias of Org’

1. sfdx force:auth:web:login -r https://test.salesforce.com -a FullSandbox
2. sfdx force:auth:web:login -r https://test.salesforce.com -a DevSandbox

* If you create an alias for the sandbox (-a option), you can reference it by this alias instead of its long and often unintuitive username.
* -r, --instanceurl INSTANCEURL the login URL of the instance the org lives on

1. sfdx force:org:list

* Allows you to see all orgs logged into
  + –verbose shows more details

1. sfdx force:project:create -n ‘name of project’

* This creates a folder that has the structure of a new salesforce project.

1. sfdx force:org:create -s -f config/project-scratch-def.json -a ‘alias of scratch org’

* ‘-s’ option indicates that you want this scratch org to be the default org for this project
* ‘-f’ option is the path to the project scratch org configuration file

1. sfdx force:user:permset:assign -n ‘Name of Permission Set’

* This will assign the current user a permission set

1. sfdx force:source:pull

* This will sync the changes made in the scratch org to local project

1. sfdx force:source:push

* will push local code to the scratch org
* add ‘-u Alias Name’ to push to a specific scratch org

1. sfdx force:data:tree:export -q "Place Query Here" -d ./data

* You can export data into a JSON format. Populate the query and location of where data should be stores
* This allows data to be moved between multiple scratch orgs for testing

1. sfdx force:data:tree:import --sobjecttreefiles data/Account.json

* This command allows you to import data into a scratch org by specifying the location of the data
* ‘-u + Alias Name’ to deploy to specific environment

1. sfdx force:apex:class:create -n ‘File Name’-d ‘File Destination’

* This will create an apex file
* Need to run this at the root of the project

1. sfdx force:lightning:component:create -n ‘File Name’ -d ‘File Destination’

* This will create a lightning component

1. sfdx force:org:display -u ‘Alias Name’

* This will display keys and their values for an org

1. sfdx force:user:password:generate -u ‘Alias Name’

* will generate a password for the org specified
* used for installing packages and such

1. sfdx force:mdapi:retrieve -s -r ./’New Folder To Hold Data’ -p ‘Project Name’ -u ‘Alias Name’-w 10

* This will retrieve the contents of the project package into the folder created/specified
* You can unzip this folded to use it to push source to new Scratch Org
  + unzip unpackaged.zip -d .

1. sfdx force:mdapi:convert -r ‘Folder Containing API Package’/

* This will convert the package that was retrieved into a workable version for developersclear
* Mdapi to source

1. sfdx force:source:convert -d ‘Folder Location’/

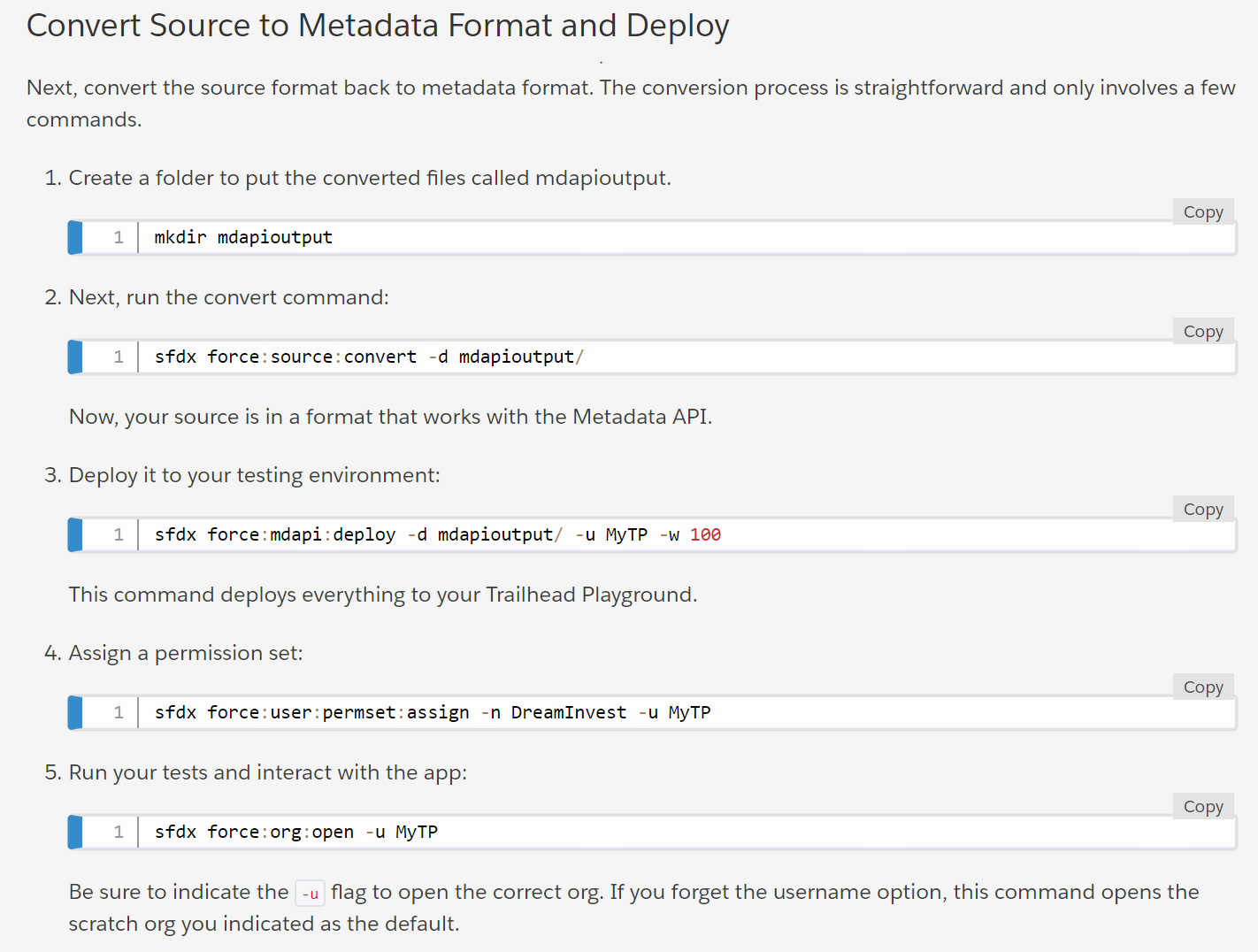
* This will convert the source to Metadata API
* Used to push code back to other Scratch Orgs

1. sfdx force:mdapi:deploy -d ‘Folder Location’/ -u ‘Alias Name’ -w 100

* This will deploy the source you converted to the org specified

**Images:**

* 19, 20



* 17, 18



**More Commands**:

-----------------------------------------------------------------------------------------------------

Contributor: Miles Sollinger

-- Authenticate in a new sandbox

sfdx force:auth:web:login -r https://force-dsc--oc02test.cs32.my.salesforce.com -a myOrg

-- Retrieve a change set from a sandbox

sfdx force:mdapi:retrieve -u myOrg -p package\_name -r ./target\_dir

-- Retrieve a package using package.xml from a sandbox

sfdx force:mdapi:retrieve -u myOrg -r ./target\_dir -k ./pkg\_dir/package.xml

-- Convert package from mdapi format to DX format

sfdx force:mdapi:convert -r ./source\_dir

-- Validate via MDAPI to sandbox

sfdx force:mdapi:deploy -d ./deploy\_dir -u targetOrg -c -w 30 -l RunLocalTests

-- Convert package from DX format to MDAPI format

sfdx force:source:convert -d ./output\_dir

-- Create a scratch org

sfdx force:org:create -f ./config/project-scratch-def.json -a wizOrg

-- Assign Modify All permission set

sfdx force:user:permset:assign -n OC02\_MTR\_Modify\_All -u wizOrg

-- Run apex tests

sfdx force:apex:test:run -s OC02Tests -w 30 -u OC02Test -c -r human

-----------------------------------------------------------------------------------------------------

sfdx force:doc:commands:list

=== Commands

force:alias:list # list username aliases for sfdx

force:alias:set # set username aliases for sfdx

force:apex:class:create # create an apex class

force:apex:execute # execute anonymous apex code

force:apex:log:get # fetch a debug log

force:apex:log:list # list debug logs

force:apex:test:report # display test results

force:apex:test:run # invoke apex tests

force:auth:jwt:grant # authorize an org using the jwt flow

force:auth:sfdxurl:store # authorize an org using an sfdx auth url

force:auth:web:login # authorize an org using the web login flow

force:config:get # get config var value(s) for given name(s)

force:config:list # list config vars for sfdx

force:config:set # set config vars for sfdx

force:data:bulk:delete # bulk delete records from a csv file

force:data:bulk:status # view the status of a bulk data load job or batch

force:data:bulk:upsert # bulk upsert records from a csv file

force:data:record:create # create a record

force:data:record:delete # delete a record

force:data:record:get # view a record

force:data:record:update # update a record

force:data:soql:query # execute a soql query

force:data:tree:export # export data from an org into sobject tree format for force:data:tree:import consumption

force:data:tree:import # import data into an org using sobject tree api

force:doc:commands:display # display help for force commands

force:doc:commands:list # list the force commands

force:lightning:app:create # create a lightning app

force:lightning:component:create # create a lightning component

force:lightning:event:create # create a lightning event

force:lightning:interface:create # create a lightning interface

force:lightning:lint # Linter for Lightning Components

force:lightning:test:create # create a lightning test

force:lightning:test:run # invoke lightning component tests

force:limits:api:display # display current org’s limits

force:mdapi:convert # convert metadata api source into the sfdx source format

force:mdapi:deploy # deploys metadata to an org using metadata api

force:mdapi:retrieve # retrieves metadata from an org using metadata api

force:org:create # create a scratch org

force:org:delete # mark a scratch org for deletion

force:org:display # get org description

force:org:list # list all active orgs you’ve created or authenticated to

force:org:open # open an org in your browser

force:package1:version:create # create a new package version in the release org

force:package1:version:create:get # retrieve status of package upload request

force:package1:version:display # display details about a package version

force:package1:version:list # list package versions for the specified package or for the org

force:package2:create # create a package2

force:package2:installed:list # list the org’s installed subscriber package2 versions

force:package2:list # list all package2 packages in the dev hub org

force:package2:manifest:create # create a manifest for the package2 from the specified directory

force:package2:members:list # list all subscriber package2 members in the org

force:package2:version:create # create a package2 version in the dev hub org

force:package2:version:create:get # retrieve a package2 version creation request in the dev hub org

force:package2:version:create:list # list package2 version creation requests in the dev hub org

force:package2:version:get # retrieve a package version in the dev hub org

force:package2:version:install # install a subscriber package2 version

force:package2:version:list # list all package2 versions in the dev hub org

force:package2:version:uninstall # uninstall a subscriber package2 version

force:package2:version:update # update a package2 version in the dev hub org

force:package:install # install a package in the target org

force:package:install:get # retrieve status of package install request

force:project:create # create a new SFDX project

force:project:upgrade # update project config files to the latest format

force:schema:sobject:describe # describe an object

force:schema:sobject:list # list all objects of a type

force:source:convert # convert sfdx source into the metadata api source format

force:source:open # edit a lightning page with lightning app builder

force:source:pull # pull source from the scratch org to the project

force:source:push # push source to an org from the project

force:source:status # list local changes and/or changes in a scratch org

force:user:password:generate # generate a password for a scratch org

force:user:permset:assign # assign a permission set to the admin user of an org

force:visualforce:component:create # create a visualforce component

force:visualforce:page:create # create a visualforce page