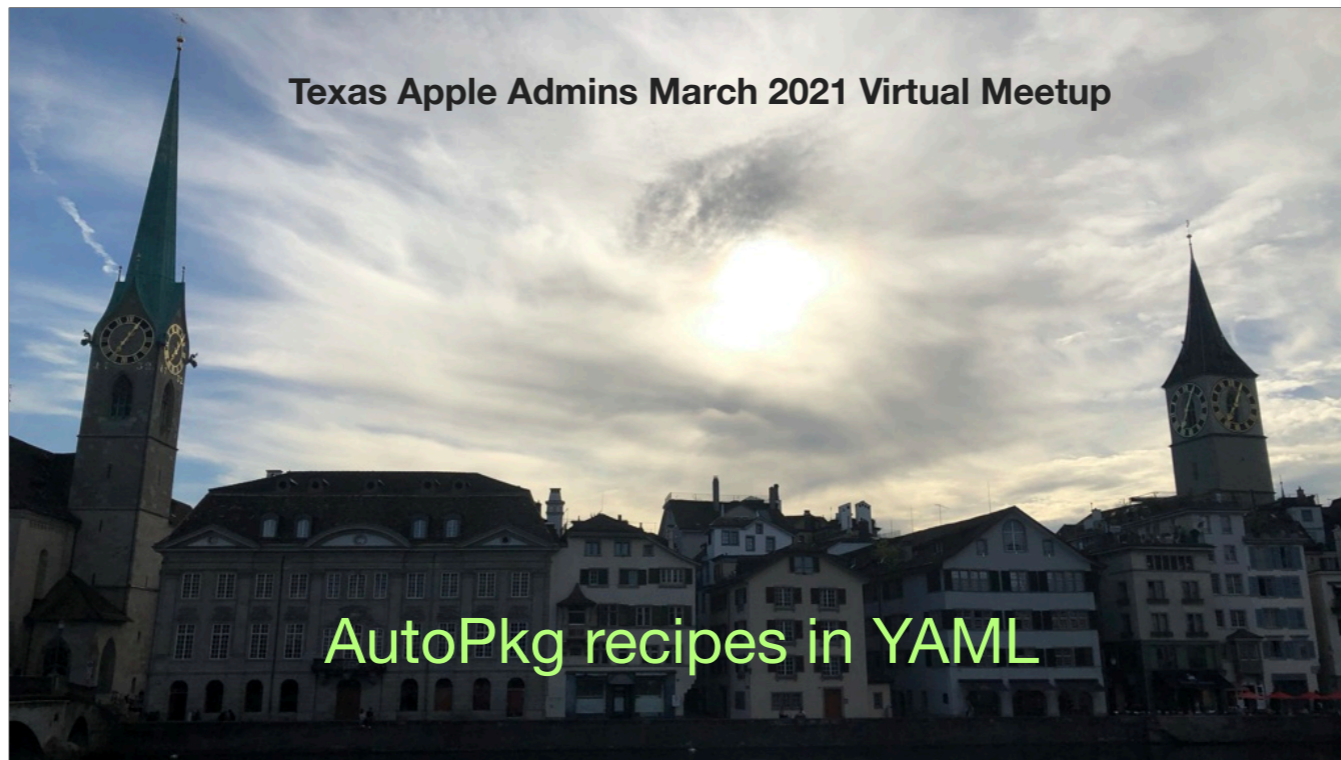


Texas Apple Admins March 2021 Virtual Meetup

AutoPkg recipes in YAML





Graham Pugh

Senior Client Engineer - Apple Services
ETH Zürich



@GrahamRPugh



@GrahamRPugh



@grahampugh



grahampugh.com

My name is Graham, I'm from the UK, and I work in Switzerland at a technical university called ETH Zürich.

I'm going to talk briefly today about how you can write AutoPkg recipes in YAML.

I've been using AutoPkg for about 7 or 8 years, writing PLIST based recipes almost from the beginning, and for the past 3 years I have been writing my AutoPkg recipes in YAML, and then converting them to PLIST for use.

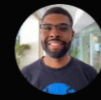
But as of AutoPkg version 2.3, thanks to the work of Elliot Jordan, YAML-formatted recipes can be run directly from AutoPkg.



It's fair to say that YAML has a lot of haters.
The Suez meme has made it to YAML haters.



Read this out



David Fowler 🇺🇸 @davidfowl · 7h

Never, we're never going to do **yaml** project files. They say never say never, but I'm saying it here: never



Read this out



Gerald Mücke @gmuecke · 20h

best tool for dealing with **YAML** files already exists:

rm



Read this out

munkireport - 30 Jun 2020



matx 21:26

Yaml 🤖

Pause

munkireport-dev - 20 May 2020



joncrain 19:36

Haven't found a **yaml** version I like

Pause

#anzmac - 13 May 2020



bartreardon 04:44

yaml can die in a hole

Pause

laa-pub - 5 Mar



macmule 11:00

there is only one thing i dislike more than **Yaml**.. marzipan

My favourite from Ben Toms (read out)

Ansible

```
---  
- name: Deploy services  
  hosts: all  
  tasks:  
    - name: Read the tasks.yml file to find what to do  
      include: tasks.yml  
    vars:  
      package: samba  
      service: smb  
      state: started  
      register: output  
  
- name: debugs the included tasks
```

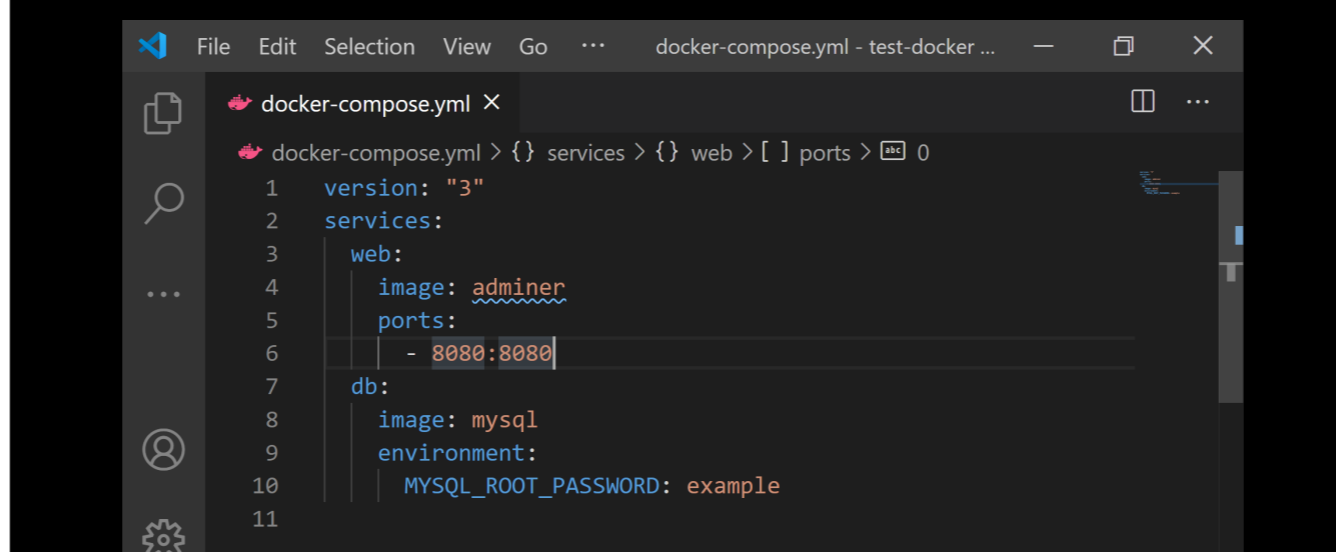
However, YAML is a widely used markup language, especially with configuration tools.

Kubernetes

```
apiVersion: v1
kind: Service
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  # if your cluster supports it, uncomment the following to automatically create
  # an external load-balanced IP for the frontend service.
  loadBalancerIP: 52.179.14.59
  type: LoadBalancer
  ports:
  - port: 80
  selector:
    app: guestbook
    tier: frontend
```

This is due to the simplicity and ease of readability.

Docker

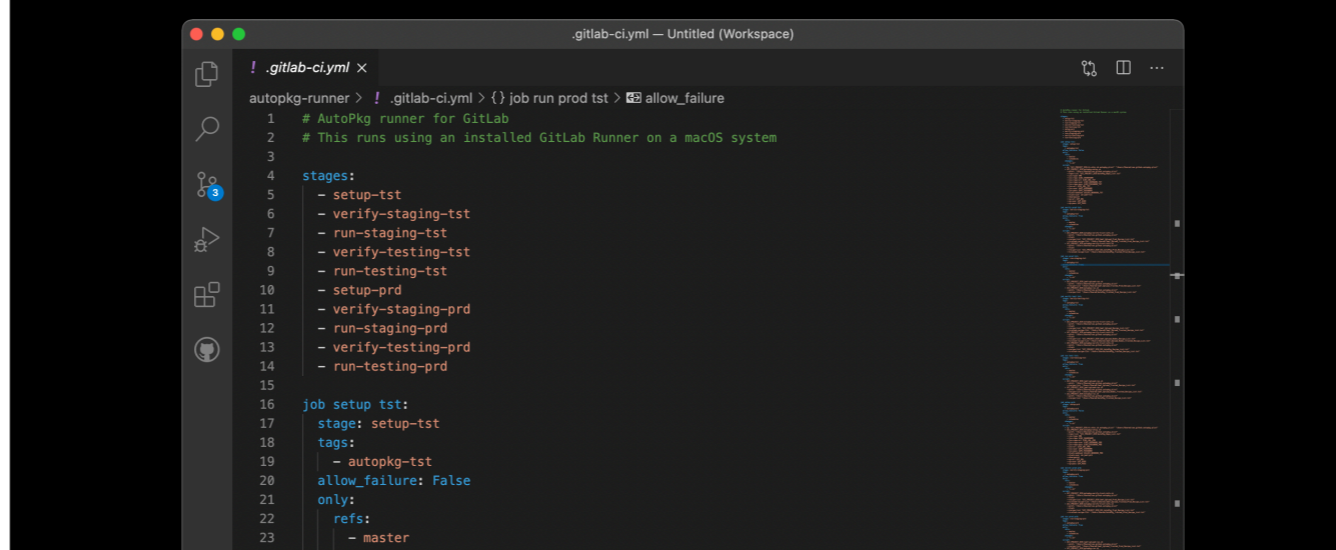
A screenshot of a code editor window showing a Docker Compose configuration file named 'docker-compose.yml'. The editor has a dark theme and a sidebar on the left with icons for file explorer, search, and settings. The main area displays the following YAML code:

```
1 version: "3"
2 services:
3   web:
4     image: adminer
5     ports:
6     - 8080:8080
7   db:
8     image: mysql
9     environment:
10    MYSQL_ROOT_PASSWORD: example
11
```

The code is color-coded, with 'version' in blue, 'services' in green, and 'web' and 'db' in blue. The 'image' and 'ports' keys are in green, and the values are in orange. The 'environment' key is in green, and its value is in orange. The line numbers 1 through 11 are on the left side of the code.

YAML essentially has the same data structure as JSON and PLIST, but instead of using different types of bracket like JSON, or markup like XML, it mainly just uses indentation.

GitLab Runner/GitHub Actions



```
.gitlab-ci.yml x
! .gitlab-ci.yml > {} job run prod tst > allow_failure
1 # AutoPkg runner for GitLab
2 # This runs using an installed GitLab Runner on a macOS system
3
4 stages:
5 - setup-tst
6 - verify-staging-tst
7 - run-staging-tst
8 - verify-testing-tst
9 - run-testing-tst
10 - setup-prd
11 - verify-staging-prd
12 - run-staging-prd
13 - verify-testing-prd
14 - run-testing-prd
15
16 job setup tst:
17 stage: setup-tst
18 tags:
19 - autopkg-tst
20 allow_failure: False
21 only:
22 refs:
23 - master
24 - schedule
```

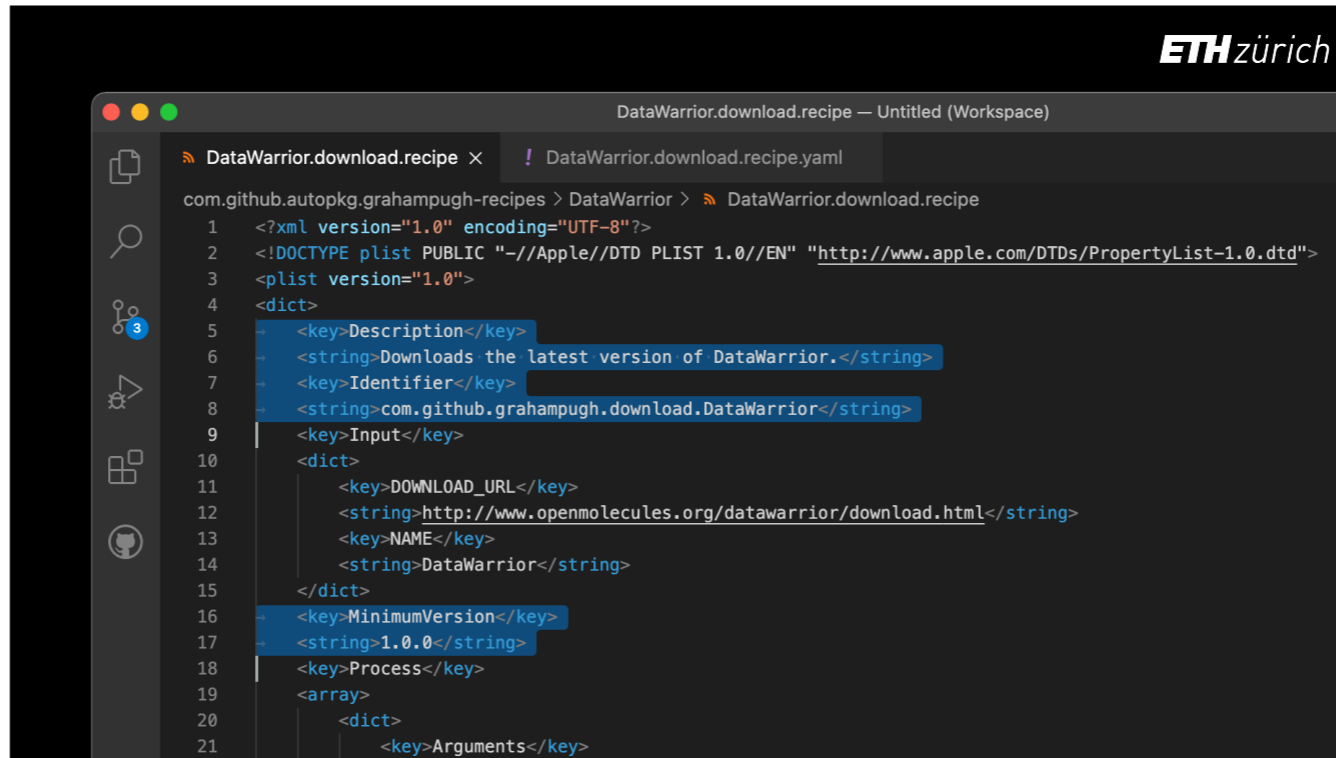
This makes it much more clear to read.

It's the same design concept as python.

And some people hate Python and YAML, because dealing with whitespace and indentation does require discipline and the right text editor.



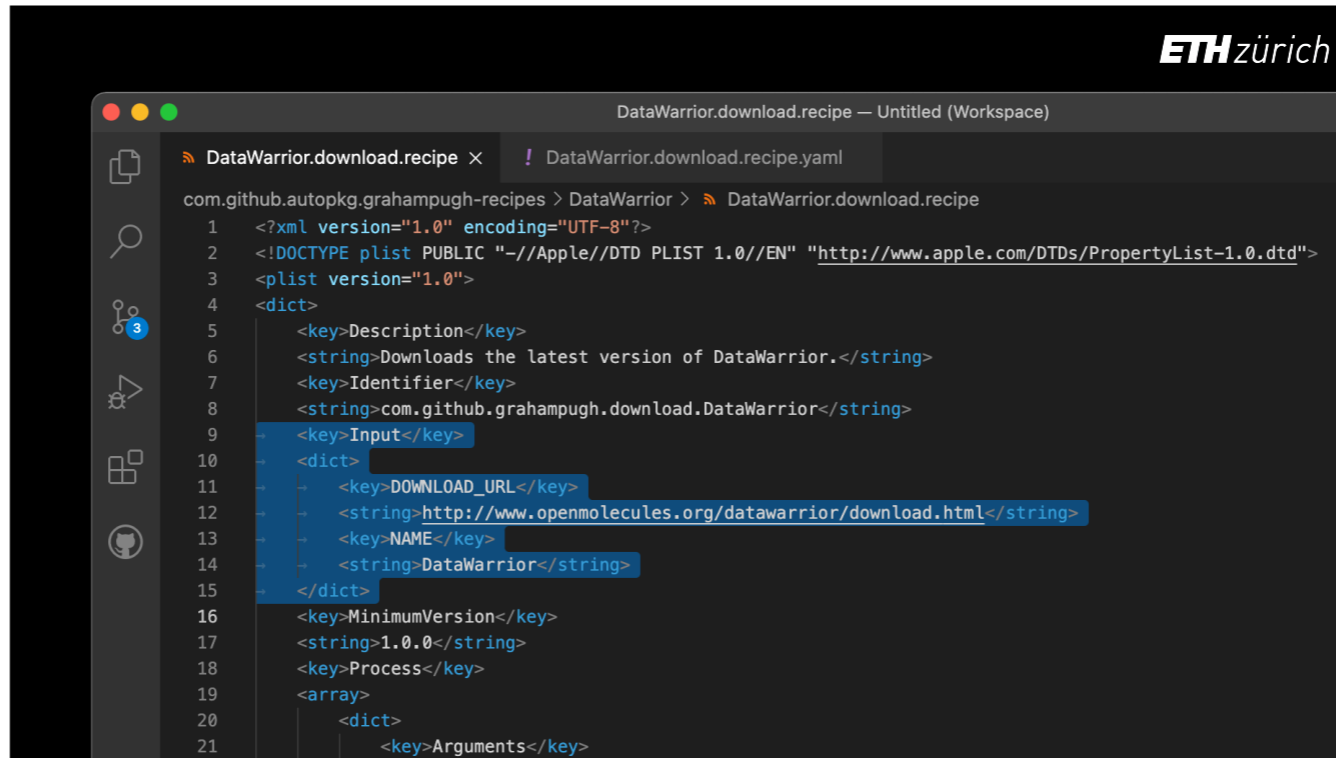
But if you are already familiar with dealing with whitespace through python programming or writing YAML config files, or if you're just willing to give marzipan another try, writing AutoPkg recipes in YAML might make sense for you.



```
com.github.autopkg.grahampugh-recipes > DataWarrior > DataWarrior.download.recipe
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
3 <plist version="1.0">
4 <dict>
5   <key>Description</key>
6   <string>Downloads the latest version of DataWarrior.</string>
7   <key>Identifier</key>
8   <string>com.github.grahampugh.download.DataWarrior</string>
9   <key>Input</key>
10  <dict>
11    <key>DOWNLOAD_URL</key>
12    <string>http://www.openmolecules.org/datawarrior/download.html</string>
13    <key>NAME</key>
14    <string>DataWarrior</string>
15  </dict>
16  <key>MinimumVersion</key>
17  <string>1.0.0</string>
18  <key>Process</key>
19  <array>
20    <dict>
21      <key>Arguments</key>
```

Let's look at a simple AutoPkg recipe written in Plist format.

- What we see are key name/value pairs written over two lines, such as the Description, Identifier and MinimumVersion keys.
- The types of value are indicated by the markup. In AutoPkg recipes this is almost always string or boolean (true/false).



```
com.github.autopkg.grahampugh-recipes > DataWarrior > DataWarrior.download.recipe
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
3 <plist version="1.0">
4 <dict>
5   <key>Description</key>
6   <string>Downloads the latest version of DataWarrior.</string>
7   <key>Identifier</key>
8   <string>com.github.grahampugh.download.DataWarrior</string>
9   <key>Input</key>
10  <dict>
11    <key>DOWNLOAD_URL</key>
12    <string>http://www.openmolecules.org/datawarrior/download.html</string>
13    <key>NAME</key>
14    <string>DataWarrior</string>
15  </dict>
16  <key>MinimumVersion</key>
17  <string>1.0.0</string>
18  <key>Process</key>
19  <array>
20    <dict>
21      <key>Arguments</key>
```

- Other values include:
- dictionaries, such as the Input dictionary.
- This is an unordered list of keys and their values.

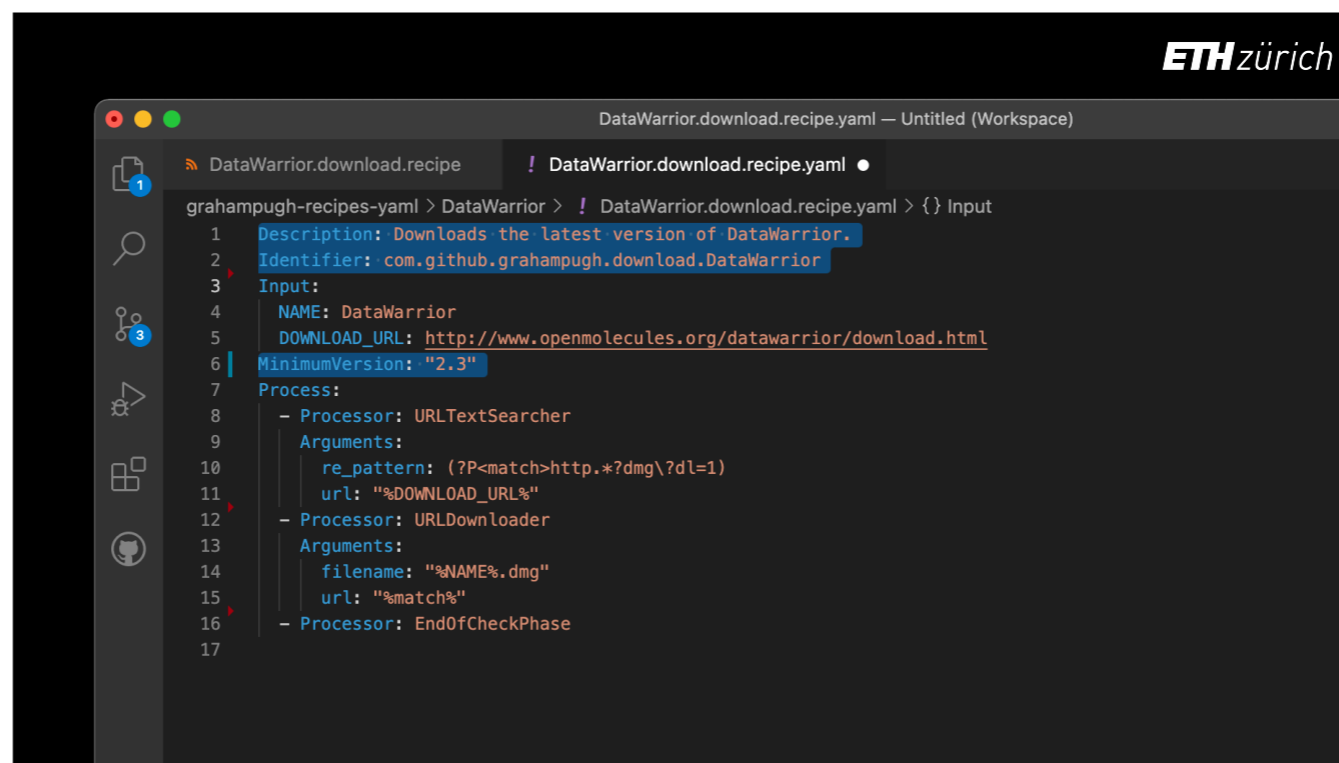
```
17 <string>1.0.0</string>
18 <key>Process</key>
19 <array>
20 <dict>
21 <key>Arguments</key>
22 <dict>
23 <key>re_pattern</key>
24 <string>(?P<match>http.*?dmg\?dl=1)</string>
25 <key>url</key>
26 <string>%DOWNLOAD_URL%</string>
27 </dict>
28 <key>Processor</key>
29 <string>URLTextSearcher</string>
30 </dict>
31 <dict>
32 <key>Arguments</key>
33 <dict>
34 <key>filename</key>
35 <string>%NAME%.dmg</string>
36 <key>url</key>
37 <string>%match%</string>
38 </dict>
```

And

- arrays, which are an ordered list.
- In this case the Process array is an ordered list of dictionaries. In AutoPkg this has to be ordered as each Processor is run in a sequence.

```
EndNoteUpdates.pkg.recipe — Untitled (Workspace)
DataWarrior.download.recipe | DataWarrior.download.recipe.yaml | EndNoteUpdates.pkg.recipe x
com.github.autopkg.grahampugh-recipes > EndNoteUpdates > EndNoteUpdates.pkg.recipe
164     <string>purge_ds_store</string>
165     <key>pkgname</key>
166     <string>%NAME%MAJOR_VERSION%-updater-%version%</string>
167     <key>pkgroot</key>
168     <string>pkgroot</string>
169     <key>scripts</key>
170     <string>Scripts</string>
171   </dict>
172 </dict>
173 <key>Processor</key>
174 <string>PkgCreator</string>
175 </dict>
176 <dict>
177   <key>Arguments</key>
178   <dict>
179     <key>path_list</key>
180     <array>
181       <string>%RECIPE_CACHE_DIR%/pkgroot</string>
182       <string>%RECIPE_CACHE_DIR%/Scripts</string>
183     </array>
184   </dict>
</dict>
```

Occasionally you will also see arrays of strings, such as like here in the PathDeleter processor, where the recipe is required to remove files and folders in a particular sequence.

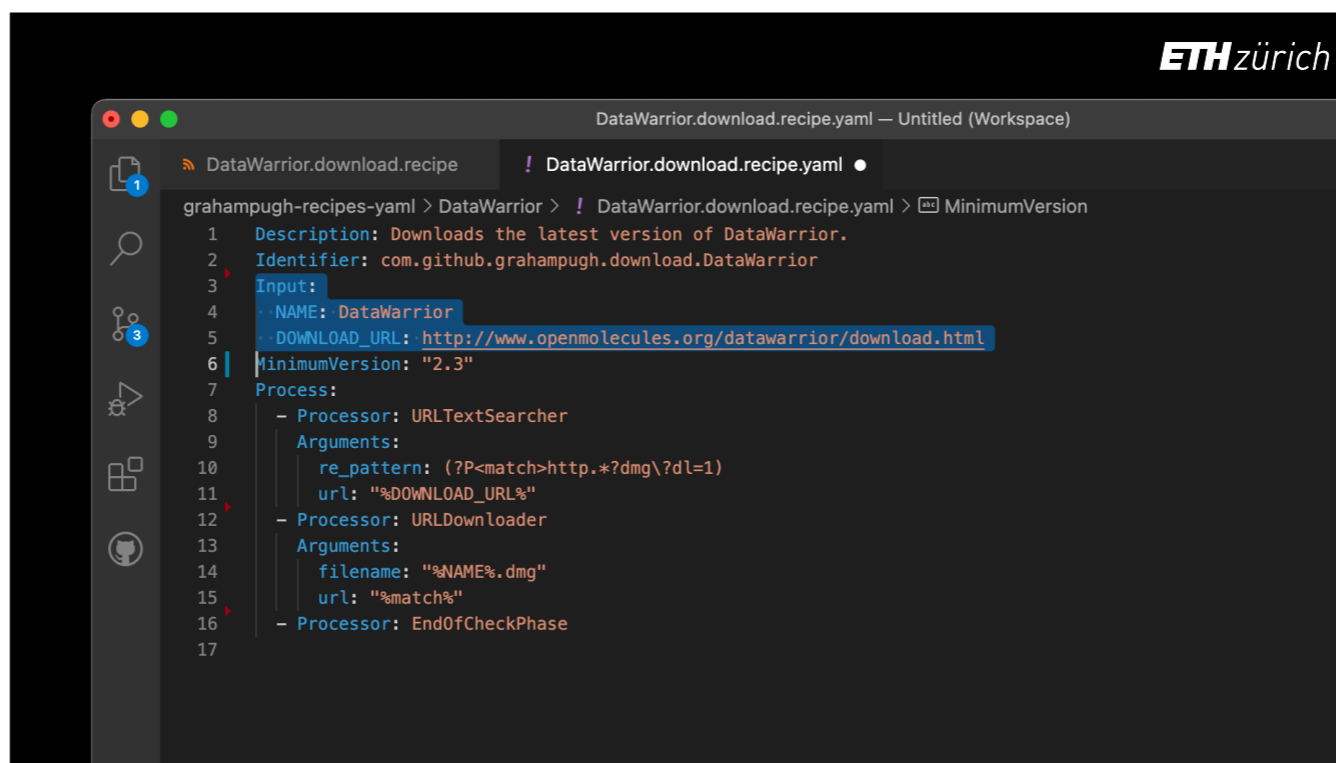


```
ETH zürich
DataWarrior.download.recipe.yaml — Untitled (Workspace)
DataWarrior.download.recipe ! DataWarrior.download.recipe.yaml ●
grahampugh-recipes-yaml > DataWarrior > ! DataWarrior.download.recipe.yaml > {} Input
1 Description: Downloads the latest version of DataWarrior.
2 Identifier: com.github.grahampugh.download.DataWarrior
3 Input:
4   NAME: DataWarrior
5   DOWNLOAD_URL: http://www.openmolecules.org/datawarrior/download.html
6   MinimumVersion: "2.3"
7 Process:
8   - Processor: URLTextSearcher
9     Arguments:
10    re_pattern: (?P<match>http.*?dmg\?dl=1)
11    url: "%DOWNLOAD_URL%"
12  - Processor: URLDownloader
13    Arguments:
14    filename: "%NAME%.dmg"
15    url: "%match%"
16  - Processor: EndOfCheckPhase
17
```

Now let's compare that with a recipe in a YAML format.
You can see instantly the first benefit of using YAML – the recipe is much shorter.

In YAML format, the key name comes before a colon and a space. The value after the colon and space.
Type is inferred.

- If the value is clearly not an integer, floating number or boolean, it is interpreted as a string.
- This means that you occasionally have to take care to specify something as a string by putting it in quotes, such as here where the MinimumVersion is 2.3 which looks like a floating number, but AutoPkg expects to be a string.

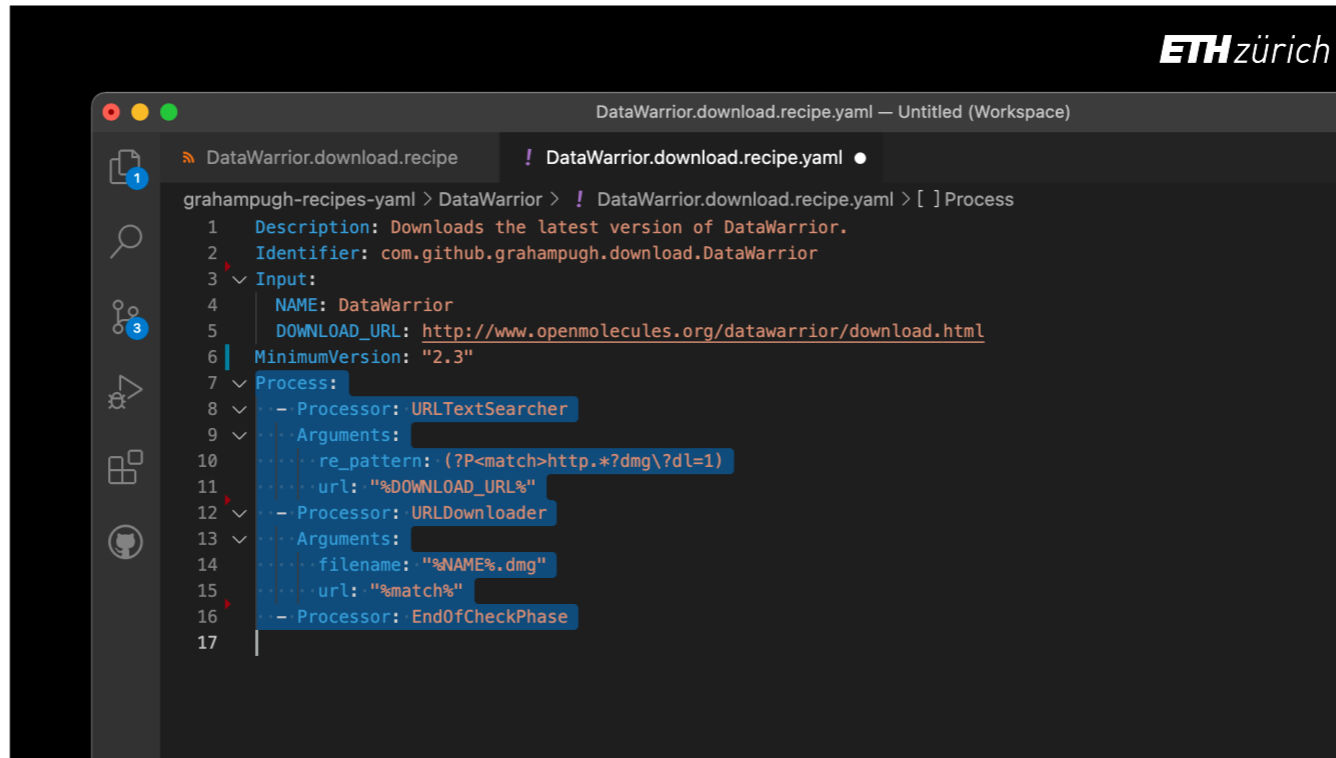


The screenshot shows a code editor window titled "DataWarrior.download.recipe.yaml — Untitled (Workspace)" with the ETH zürich logo in the top right corner. The editor displays a YAML file named "DataWarrior.download.recipe.yaml" with the following content:

```
1 Description: Downloads the latest version of DataWarrior.
2 Identifier: com.github.grahampugh.download.DataWarrior
3 Input:
4   NAME: DataWarrior
5   DOWNLOAD_URL: http://www.openmolecules.org/datawarrior/download.html
6 MinimumVersion: "2.3"
7 Process:
8   - Processor: URLTextSearcher
9     Arguments:
10      re_pattern: (?P<match>http.*?dmg\?dl=1)
11      url: "%DOWNLOAD_URL%"
12   - Processor: URLDownloader
13     Arguments:
14      filename: "%NAME%.dmg"
15      url: "%match%"
16   - Processor: EndOfCheckPhase
17
```

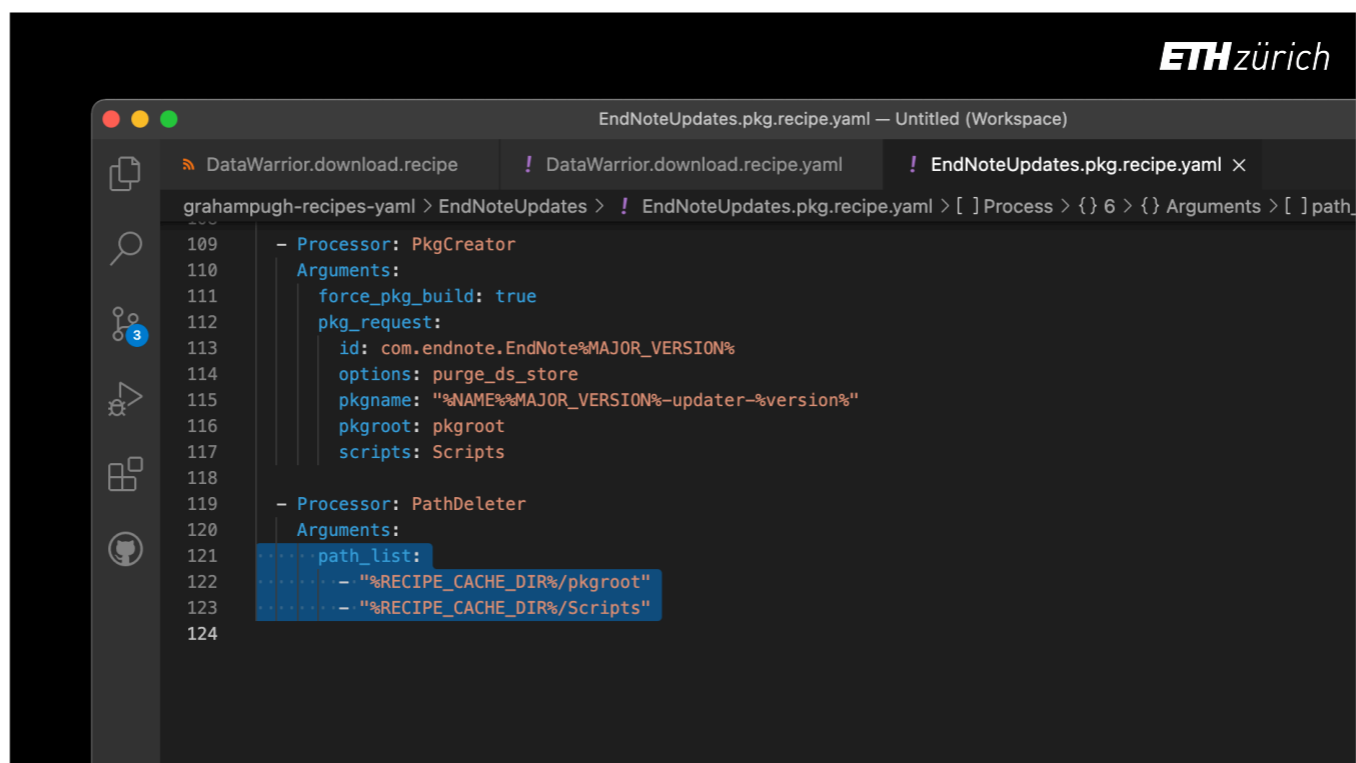
The code illustrates how a dictionary value is represented in YAML by indenting each key pair by two spaces. The "Input" section is a dictionary with two key-value pairs: "NAME" and "DOWNLOAD_URL". The "Process" section is a list of processors, each with its own set of arguments, also indented by two spaces.

A dictionary value is represented by indenting each of the key pairs in the dictionary by two spaces.



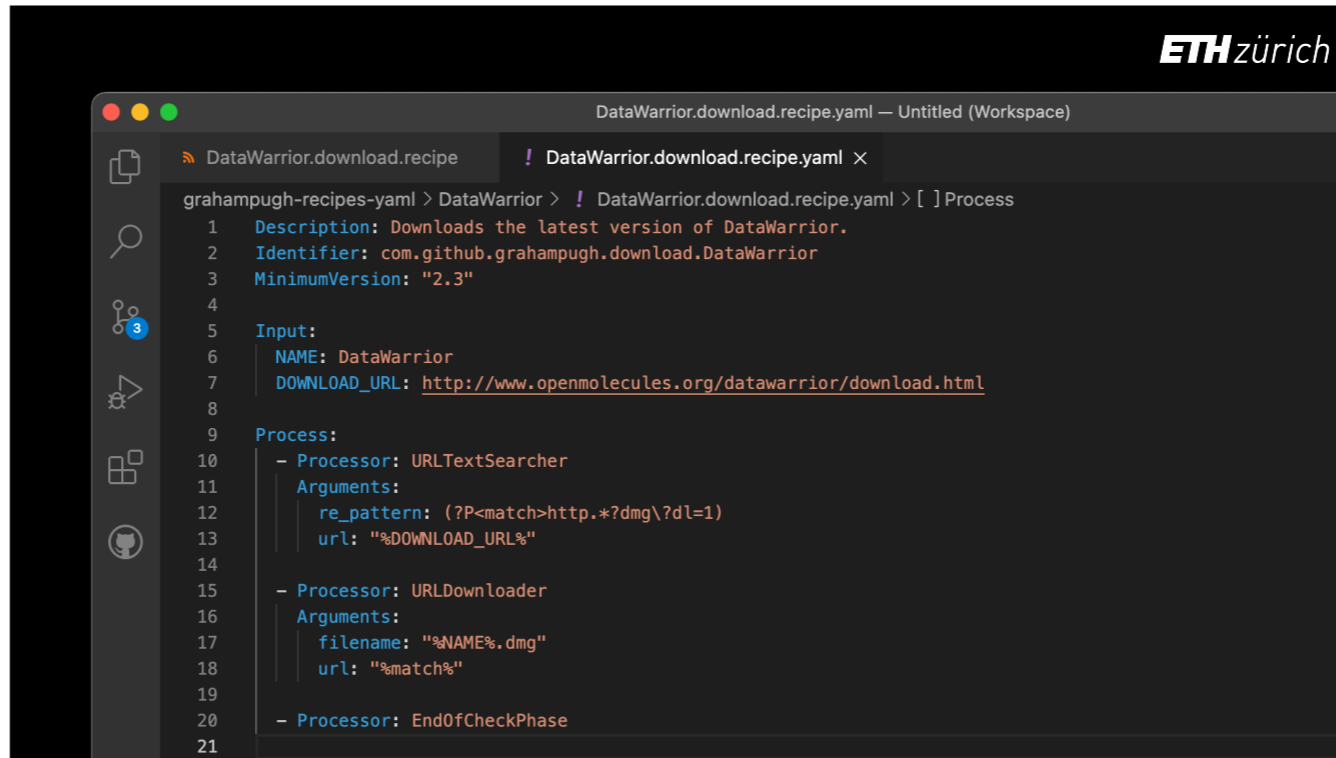
```
DataWarrior.download.recipe.yaml — Untitled (Workspace)
DataWarrior.download.recipe ! DataWarrior.download.recipe.yaml ●
grahampugh-recipes-yaml > DataWarrior > ! DataWarrior.download.recipe.yaml > [ ] Process
1 Description: Downloads the latest version of DataWarrior.
2 Identifier: com.github.grahampugh.download.DataWarrior
3 Input:
4   NAME: DataWarrior
5   DOWNLOAD_URL: http://www.openmolecules.org/datawarrior/download.html
6   MinimumVersion: "2.3"
7 Process:
8   - Processor: URLTextSearcher
9     Arguments:
10      re_pattern: (?P<match>http.*?dmg\?dl=1)
11      url: "%DOWNLOAD_URL%"
12   - Processor: URLDownloader
13     Arguments:
14      filename: "%NAME%.dmg"
15      url: "%match%"
16   - Processor: EndOfCheckPhase
17
```

Array items are represented by a dash indented by two spaces.
Here we see the 3 processor dictionaries in an array.



```
EndNoteUpdates.pkg.recipe.yaml — Untitled (Workspace)
DataWarrior.download.recipe | DataWarrior.download.recipe.yaml | EndNoteUpdates.pkg.recipe.yaml x
grahampugh-recipes-yaml > EndNoteUpdates > ! EndNoteUpdates.pkg.recipe.yaml > [ ]Process > { } 6 > { } Arguments > [ ] path_
109   - Processor: PkgCreator
110     Arguments:
111       force_pkg_build: true
112       pkg_request:
113         id: com.endnote.EndNote%MAJOR_VERSION%
114         options: purge_ds_store
115         pkgname: "%NAME%%MAJOR_VERSION%-updater-%version%"
116         pkgroot: pkgroot
117         scripts: Scripts
118
119   - Processor: PathDeleter
120     Arguments:
121       path_list:
122         - "%RECIPE_CACHE_DIR%/pkgroot"
123         - "%RECIPE_CACHE_DIR%/Scripts"
124
```

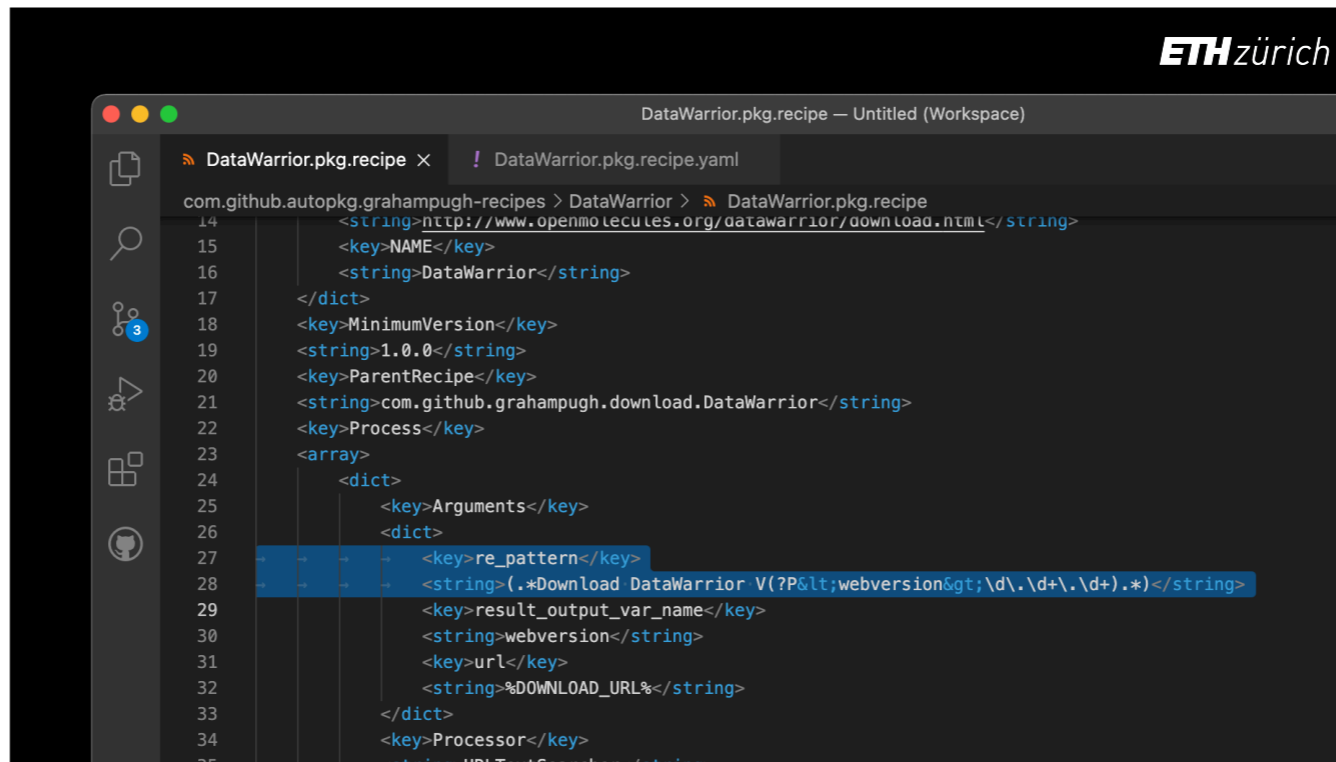
And here is that array of paths to be deleted in the PathDeleter processor.



```
grahampugh-recipes-yaml > DataWarrior > ! DataWarrior.download.recipe.yaml > [ ] Process
1  Description: Downloads the latest version of DataWarrior.
2  Identifier: com.github.grahampugh.download.DataWarrior
3  MinimumVersion: "2.3"
4
5  Input:
6    NAME: DataWarrior
7    DOWNLOAD_URL: http://www.openmolecules.org/datawarrior/download.html
8
9  Process:
10 - Processor: URLTextSearcher
11   Arguments:
12     re_pattern: (?P<match>http.*?dmg\?dl=1)
13     url: "%DOWNLOAD_URL%"
14
15 - Processor: URLDownloader
16   Arguments:
17     filename: "%NAME%.dmg"
18     url: "%match%"
19
20 - Processor: EndOfCheckPhase
21
```

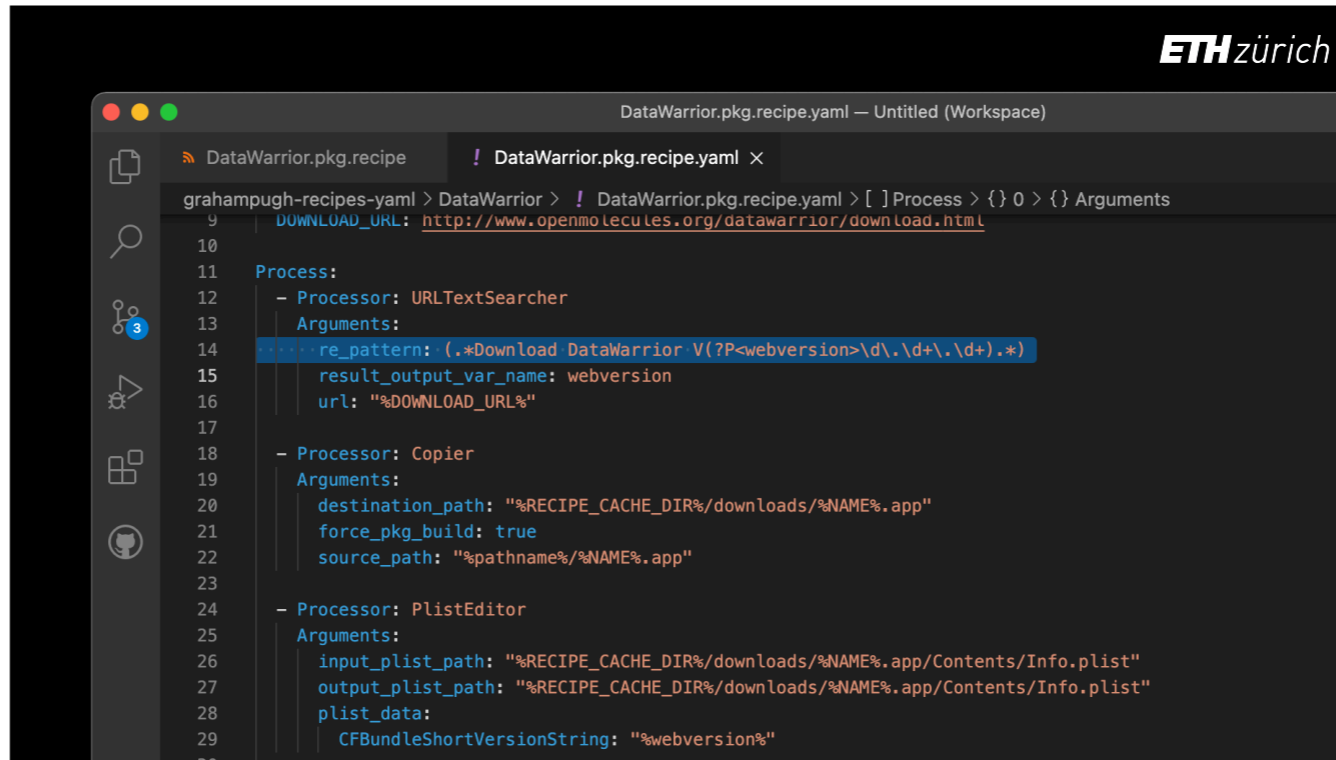
YAML doesn't care about spaces between items. The order of lists is also unimportant, except for items in arrays.

So what I do to make the YAML recipes even nicer to read is to move the Description, Identifier and Minimum Version to the top, add spaces between the Input and Process sections, and a space between each process.



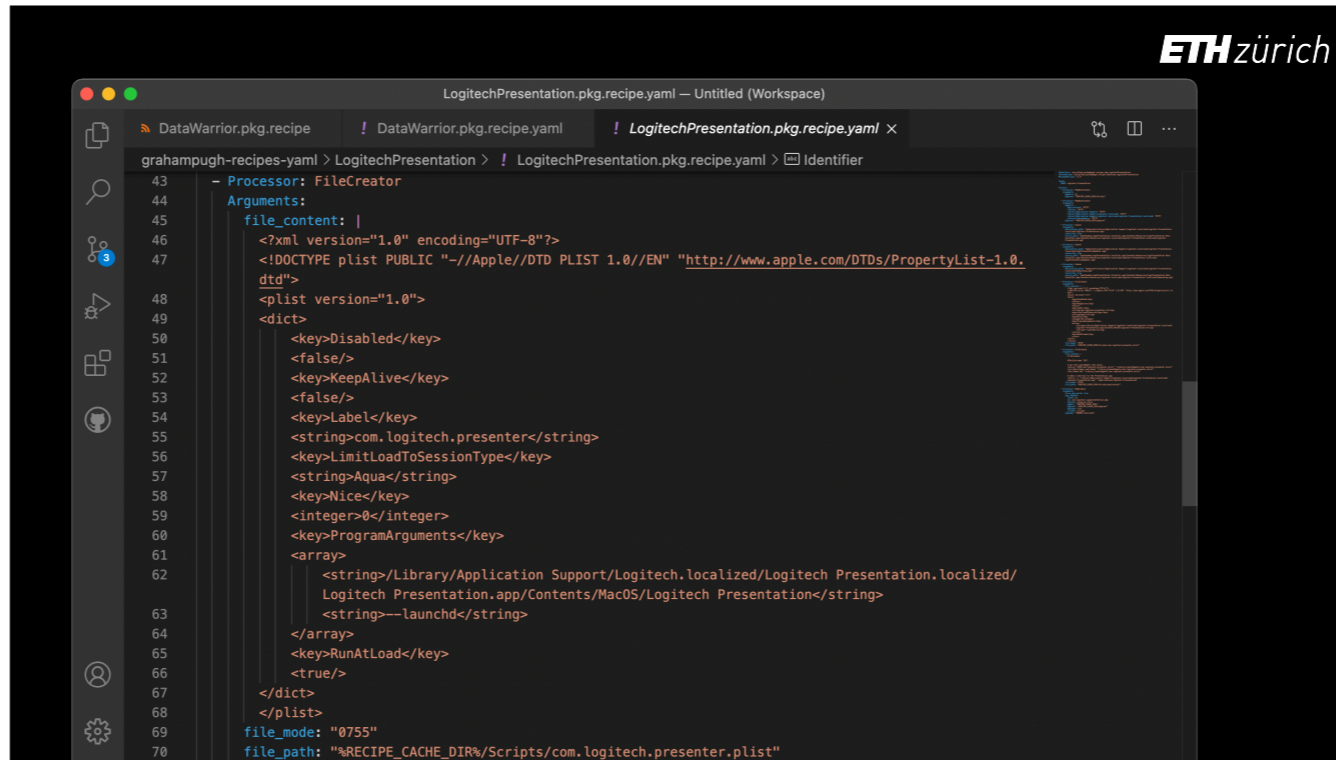
```
com.github.autopkg.grahampugh-recipes > DataWarrior > DataWarrior.pkg.recipe
14 <string>http://www.openmolecul.es.org/datawarrior/download.html</string>
15 <key>NAME</key>
16 <string>DataWarrior</string>
17 </dict>
18 <key>MinimumVersion</key>
19 <string>1.0.0</string>
20 <key>ParentRecipe</key>
21 <string>com.github.grahampugh.download.DataWarrior</string>
22 <key>Process</key>
23 <array>
24 <dict>
25 <key>Arguments</key>
26 <dict>
27 <key>re_pattern</key>
28 <string>(. *Download DataWarrior V(?P<webversion>\d+\.\d+\.\d+). *)</string>
29 <key>result_output_var_name</key>
30 <string>webversion</string>
31 <key>url</key>
32 <string>%DOWNLOAD_URL%</string>
33 </dict>
34 <key>Processor</key>
35 <string>URLTextSearcher</string>
```

Another benefit of YAML recipes over PLISTS is that fewer things need to be escaped. Here's a URLTextSearcher process with a regex pattern key in a PLIST-based recipe. We have to escape the less-than and greater-than signs in the pattern to prevent these being interpreted as XML code.



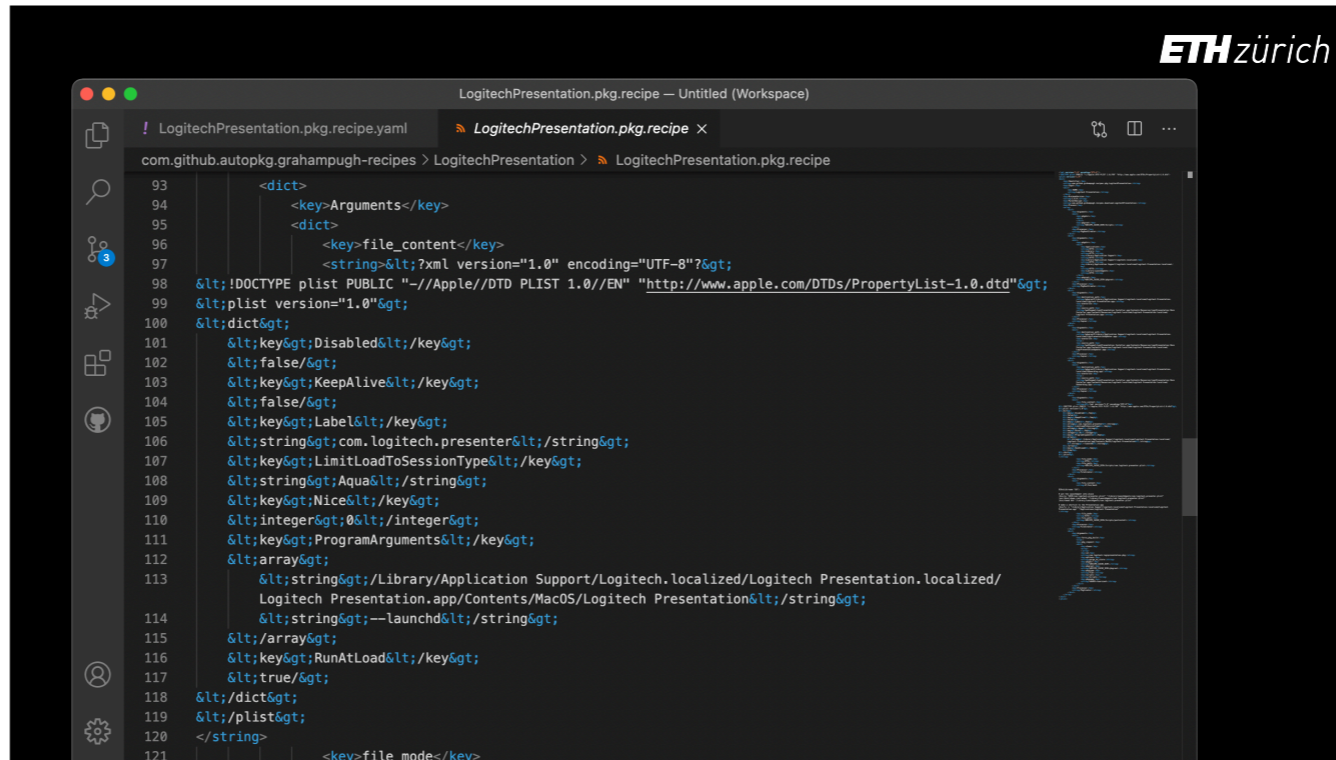
```
grahampugh-recipes-yaml > DataWarrior > ! DataWarrior.pkg.recipe.yaml > [ ] Process > {} 0 > {} Arguments
9 | DOWNLOAD_URL: http://www.openmoleculer.org/datawarrior/download.html
10 |
11 | Process:
12 |   - Processor: URLTextSearcher
13 |     Arguments:
14 |       re_pattern: (.*Download DataWarrior V(?P<webversion>\d\.\d+\.\d+).*)
15 |       result_output_var_name: webversion
16 |       url: "%DOWNLOAD_URL%"
17 |
18 |   - Processor: Copier
19 |     Arguments:
20 |       destination_path: "%RECIPE_CACHE_DIR%/downloads/%NAME%.app"
21 |       force_pkg_build: true
22 |       source_path: "%pathname%/%NAME%.app"
23 |
24 |   - Processor: PlistEditor
25 |     Arguments:
26 |       input_plist_path: "%RECIPE_CACHE_DIR%/downloads/%NAME%.app/Contents/Info.plist"
27 |       output_plist_path: "%RECIPE_CACHE_DIR%/downloads/%NAME%.app/Contents/Info.plist"
28 |       plist_data:
29 |         CFBundleShortVersionString: "%webversion%"
30 |
```

But in YAML, no escaping is necessary.



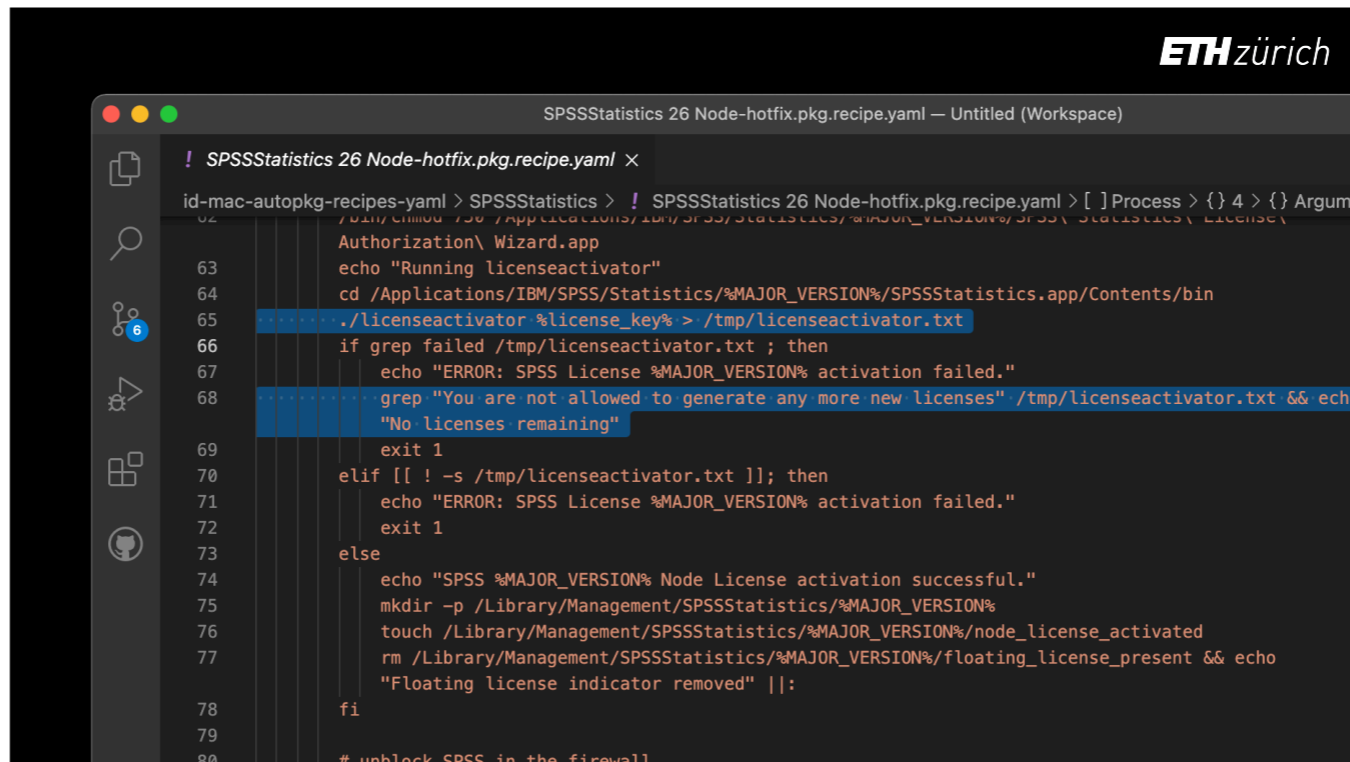
```
LogitechPresentation.pkg.recipe.yaml — Untitled (Workspace)
DataWarrior.pkg.recipe ! DataWarrior.pkg.recipe.yaml ! LogitechPresentation.pkg.recipe.yaml x
grahampugh-recipes-yaml > LogitechPresentation > ! LogitechPresentation.pkg.recipe.yaml > Identifier
43 - Processor: FileCreator
44 Arguments:
45   file_content: |
46     <?xml version="1.0" encoding="UTF-8"?>
47     <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.
48     dtd">
49     <plist version="1.0">
50     <dict>
51       <key>Disabled</key>
52       <false/>
53       <key>KeepAlive</key>
54       <false/>
55       <key>Label</key>
56       <string>com.logitech.presenter</string>
57       <key>LimitLoadToSessionType</key>
58       <string>Aqua</string>
59       <key>Nice</key>
60       <integer>0</integer>
61       <key>ProgramArguments</key>
62       <array>
63         <string>/Library/Application Support/Logitech.localized/Logitech Presentation.localized/
64         Logitech Presentation.app/Contents/MacOS/Logitech Presentation</string>
65         <string>--launchd</string>
66       </array>
67       <key>RunAtLoad</key>
68       <true/>
69     </dict>
70     </plist>
file_mode: "0755"
file_path: "%RECIPE_CACHE_DIR%/Scripts/com.logitech.presenter.plist"
```

Here's an example of a FileCreator process, where the file being written is a PLIST. Note that in YAML files, multiline string blocks can be represented by a pipe sign, followed by a line break. The contents of the string block must be indented two spaces from the key. None of the markup in the file needs to be escaped.



```
LogitechPresentation.pkg.recipe — Untitled (Workspace)
! LogitechPresentation.pkg.recipe.yaml  LogitechPresentation.pkg.recipe x
com.github.autopkg.grahampugh-recipes > LogitechPresentation > LogitechPresentation.pkg.recipe
93     <dict>
94     <key>Arguments</key>
95     <dict>
96     <key>file_content</key>
97     <string>&lt;?xml version="1.0" encoding="UTF-8"?&gt;
98     &lt;!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd"&gt;
99     &lt;plist version="1.0"&gt;
100    &lt;dict&gt;
101        &lt;key&gt;Disabled&lt;/key&gt;
102        &lt;false/&gt;
103        &lt;key&gt;KeepAlive&lt;/key&gt;
104        &lt;false/&gt;
105        &lt;key&gt;Label&lt;/key&gt;
106        &lt;string&gt;com.logitech.presenter&lt;/string&gt;
107        &lt;key&gt;LimitLoadToSessionType&lt;/key&gt;
108        &lt;string&gt;Aqua&lt;/string&gt;
109        &lt;key&gt;Nice&lt;/key&gt;
110        &lt;integer&gt;0&lt;/integer&gt;
111        &lt;key&gt;ProgramArguments&lt;/key&gt;
112        &lt;array&gt;
113            &lt;string&gt;/Library/Application Support/Logitech.localized/Logitech Presentation.localized/
114            Logitech Presentation.app/Contents/MacOS/Logitech Presentation&lt;/string&gt;
115            &lt;string&gt;--launchd&lt;/string&gt;
116        &lt;/array&gt;
117        &lt;key&gt;RunAtLoad&lt;/key&gt;
118        &lt;true/&gt;
119    &lt;/dict&gt;
120    &lt;/plist&gt;
121 </string>
    <key>file_mode</key>
```

Inevitably, this is messy and hard to read and write in a PLIST-formatted recipe...



```
ETH zürich
SPSSStatistics 26 Node-hotfix.pkg.recipe.yaml — Untitled (Workspace)
! SPSSStatistics 26 Node-hotfix.pkg.recipe.yaml x
id-mac-autopkg-recipes-yaml > SPSSStatistics > ! SPSSStatistics 26 Node-hotfix.pkg.recipe.yaml > [ ] Process > { } 4 > { } Argument
/bin/chmod 750 /Applications/IBM/SPSS/Statistics/%MAJOR_VERSION%/SPSS\ Statistics\ License\
Authorization\ Wizard.app
63 echo "Running licenseactivator"
64 cd /Applications/IBM/SPSS/Statistics/%MAJOR_VERSION%/SPSSStatistics.app/Contents/bin
65 ..... ./licenseactivator %license_key% > /tmp/licenseactivator.txt
66 if grep failed /tmp/licenseactivator.txt ; then
67     echo "ERROR: SPSS License %MAJOR_VERSION% activation failed."
68     ..... grep "You are not allowed to generate any more new licenses" /tmp/licenseactivator.txt && echo
        "No licenses remaining"
69     exit 1
70 elif [[ ! -s /tmp/licenseactivator.txt ]]; then
71     echo "ERROR: SPSS License %MAJOR_VERSION% activation failed."
72     exit 1
73 else
74     echo "SPSS %MAJOR_VERSION% Node License activation successful."
75     mkdir -p /Library/Management/SPSSStatistics/%MAJOR_VERSION%
76     touch /Library/Management/SPSSStatistics/%MAJOR_VERSION%/node_license_activated
77     rm /Library/Management/SPSSStatistics/%MAJOR_VERSION%/floating_license_present && echo
        "Floating license indicator removed" ||:
78 fi
79
80 # unblock SPSS in the firewall
```

If you are creating scripts with the FileCreator processor, things like less-than, greater-than signs and ampersands do not need to be escaped.

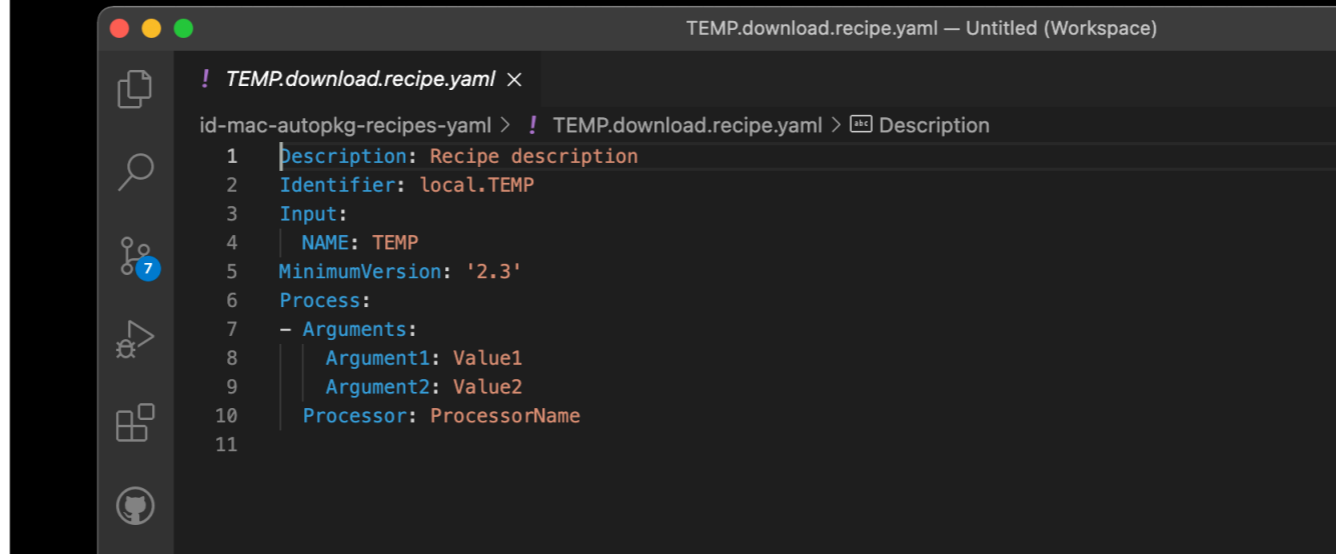
This means you can copy and paste the tested script into the YAML file, and all you have to do is indent it all to the correct level, which is easy in a good text editor like Visual Studio Code, Atom or BBEdit.

```
Fiji.pkg.recipe.yaml — Untitled (Workspace)
! Fiji.pkg.recipe.yaml x
id-mac-autopkg-recipes-yaml > Fiji > ! Fiji.pkg.recipe.yaml > [ ]Process > {} 0 > {} Arguments
preinstall script that will check for an existing Fiji.app in /Applications and remove it if found.
2 Identifier: ch.ethz.id.pkg.Fiji
3 MinimumVersion: 1.0.0
4 ParentRecipe: com.github.rtrouton.download.Fiji
5
6 Input:
7   NAME: Fiji
8
9 Process:
10  - Arguments:
11    pkgdirs:
12      Applications: '0775'
13      Library: '0775'
14      'Library/Application Support': '0775'
15      'Library/Application Support/Fiji': '0775'
16    pkgroot: '%RECIPE_CACHE_DIR%/%NAME%'
17    Processor: PkgRootCreator
18
19  - Arguments:
20    archive_path: '%pathname%'
21    destination_path: '%pkgroot%/Applications'
```

A couple of quick things to note about YAML:

- We already mentioned that some string values that look like numbers may need to be quoted. Here you see the octal permissions in the PkgRootCreator in quotes for example.
 - Occasionally you can come across key **names** that require quotes, such as when creating subfolders in the PkgRootCreator processor.
 - You also note that the AutoPkg variables with percent signs are quoted in my recipes. This isn't actually necessary as far as AutoPkg or YAML is concerned, but the text editors I've used don't seem to like it if I don't.

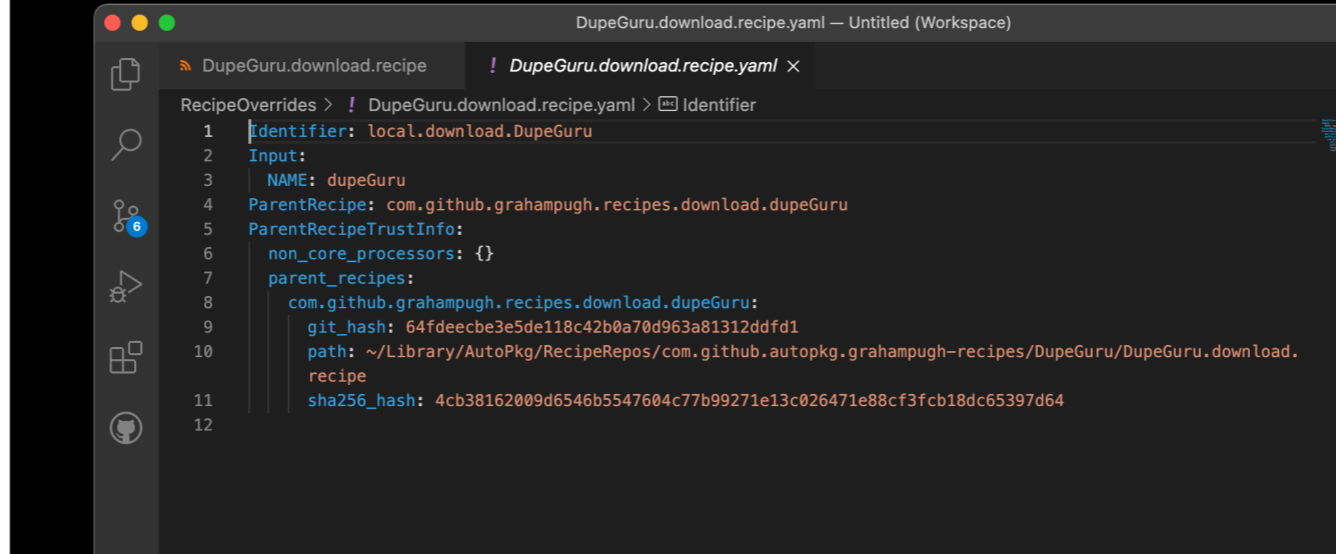
```
% autopkg new-recipe TEMP.download.recipe.yaml  
--format=yaml
```



```
TEMP.download.recipe.yaml — Untitled (Workspace)  
! TEMP.download.recipe.yaml ×  
id-mac-autopkg-recipes-yaml > ! TEMP.download.recipe.yaml > Description  
1 Description: Recipe description  
2 Identifier: local.TEMP  
3 Input:  
4   NAME: TEMP  
5   MinimumVersion: '2.3'  
6 Process:  
7   - Arguments:  
8     Argument1: Value1  
9     Argument2: Value2  
10  Processor: ProcessorName  
11
```

If you want to try writing recipes in YAML format from scratch, you can easily create a template for a new recipe with autopkg's **new-recipe** argument.

```
% autopkg make-override DupeGuru.download --format=yaml
```



The screenshot shows a code editor window titled "DupeGuru.download.recipe.yaml — Untitled (Workspace)". The editor displays a YAML file with the following content:

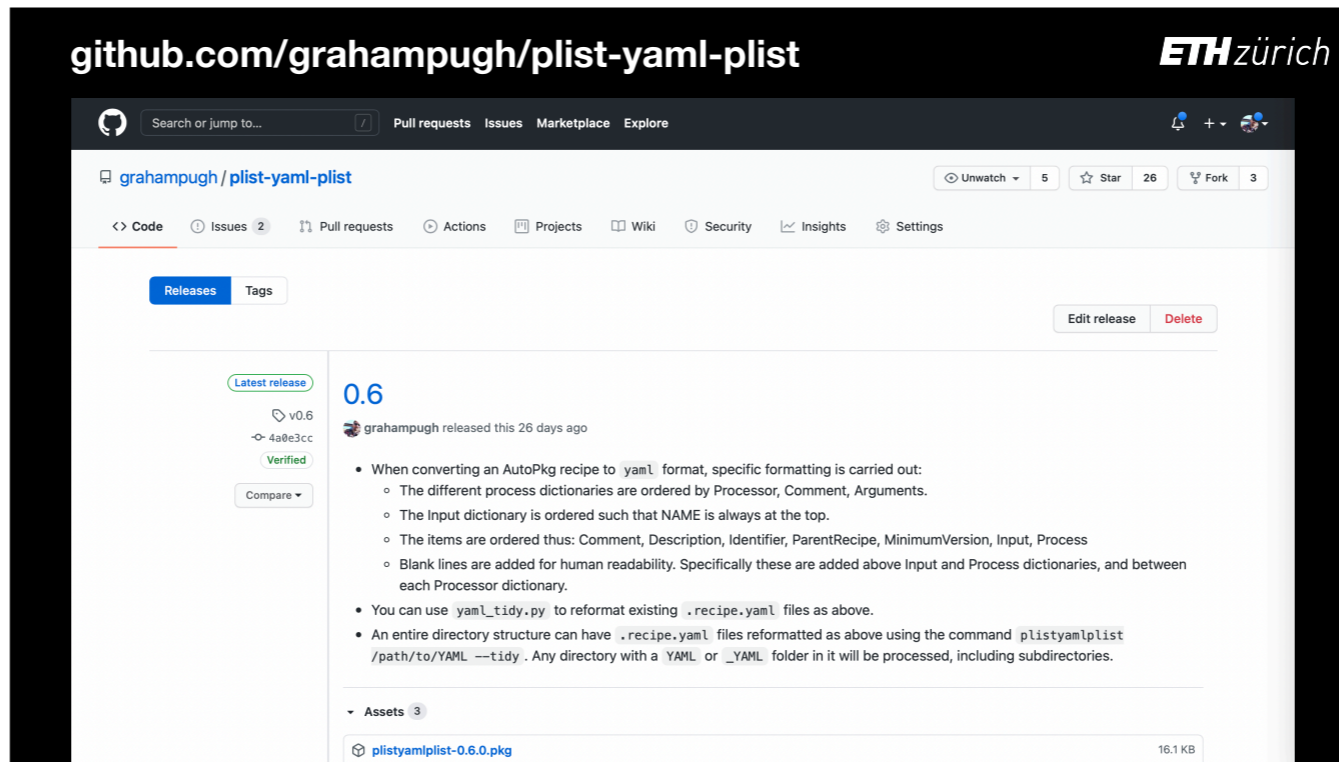
```
RecipeOverrides > ! DupeGuru.download.recipe.yaml > Identifier
1 Identifier: local.download.DupeGuru
2 Input:
3   NAME: dupeGuru
4 ParentRecipe: com.github.grahampugh.recipes.download.dupeGuru
5 ParentRecipeTrustInfo:
6   non_core_processors: {}
7   parent_recipes:
8     com.github.grahampugh.recipes.download.dupeGuru:
9       git_hash: 64fdeecbe3e5de118c42b0a70d963a81312ddfd1
10      path: ~/Library/AutoPkg/RecipeRepos/com.github.autopkg.grahampugh-recipes/DupeGuru/download.recipe
11      sha256_hash: 4cb38162009d6546b5547604c77b99271e13c026471e88cf3fcb18dc65397d64
12
```

You can also make your recipe **overrides** in YAML format, using the **format=yaml** argument.

Note that a PLIST recipe override takes precedence in the search order, so don't forget to delete your old override if you switch formats on the same device.

- YAML-based recipes must end in ***.recipe.yaml***
- ***MinimumVersion*** should be 2.3
- Parent and child recipes, and overrides, can be in different formats
- ***.recipe*** files take precedence over ***.recipe.yaml*** in the search order
- ***.recipe.plist*** is now also a valid file suffix for PLIST-based recipes

Here's some more hints...
(Explain why minimum version is 2.3)



- If you want to convert an existing AutoPkg recipe to YAML format, or indeed a YAML format recipe to PLIST, you can use a tool I made called plist-yaml-plist

```
% plistyamlplist /path/to/Some.recipe  
/path/to/Some.recipe.yaml
```

```
% plistyamlplist /path/to/Some.recipe.yaml  
/path/to/Some.recipe
```

After installing the package, you just use the `plistyamlplist` command and provide the source and destination paths. The tool will convert in both directions, based on the filenames.

There's a bunch of other options in this tool for converting entire folders and prettifying existing YAML recipes.

Conclusions

- AutoPkg version 2.3 supports YAML-format recipes
- YAML-format recipes are shorter and easier to read and write
- PLIST- and YAML-format recipes can be used interchangeably
- Marzipan is exceedingly good





That's all from me, thank you for listening!