# DBB zAppBuild – Intro

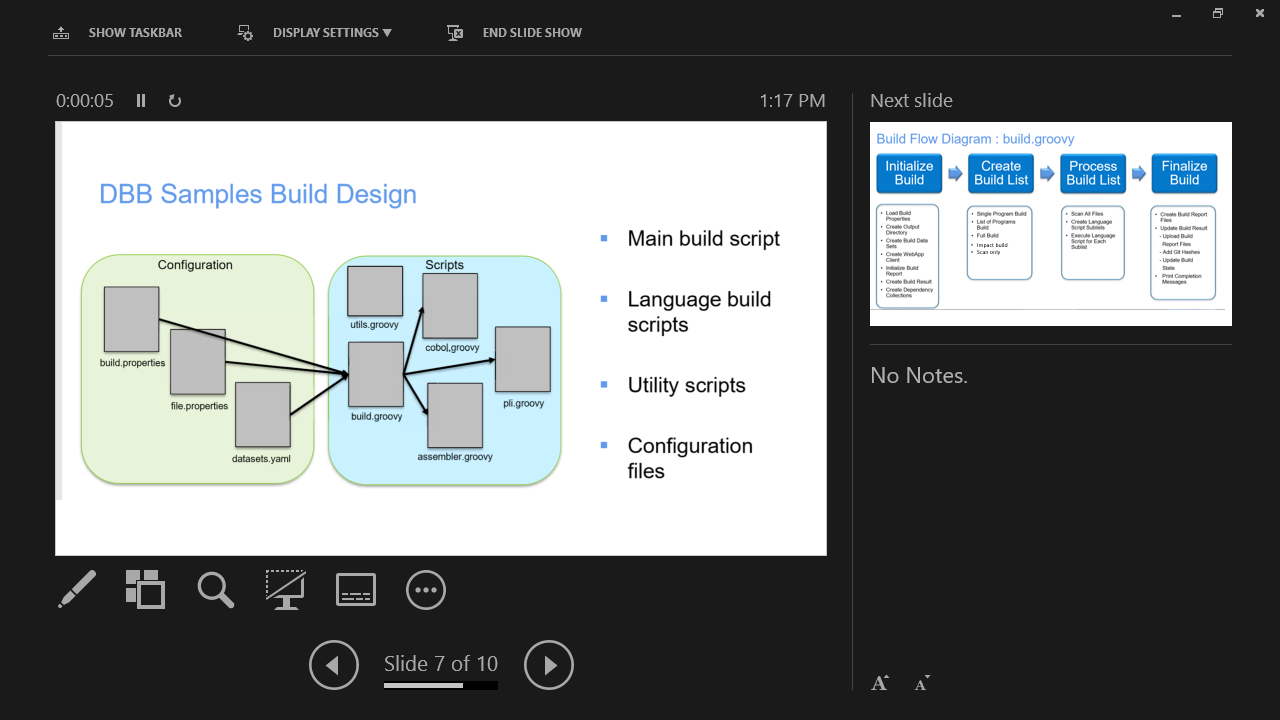
# By Nelson Lopez IBM

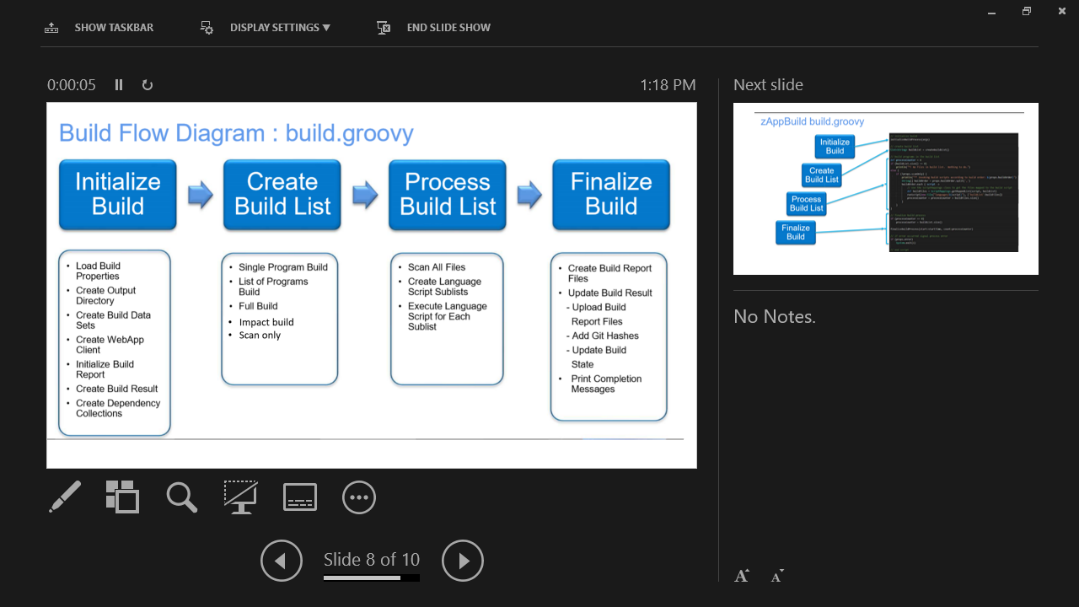
This is an introduction to the public DBB Groovy framework called zAppBuild. For more details see <https://github.com/IBM/dbb-zappbuild>

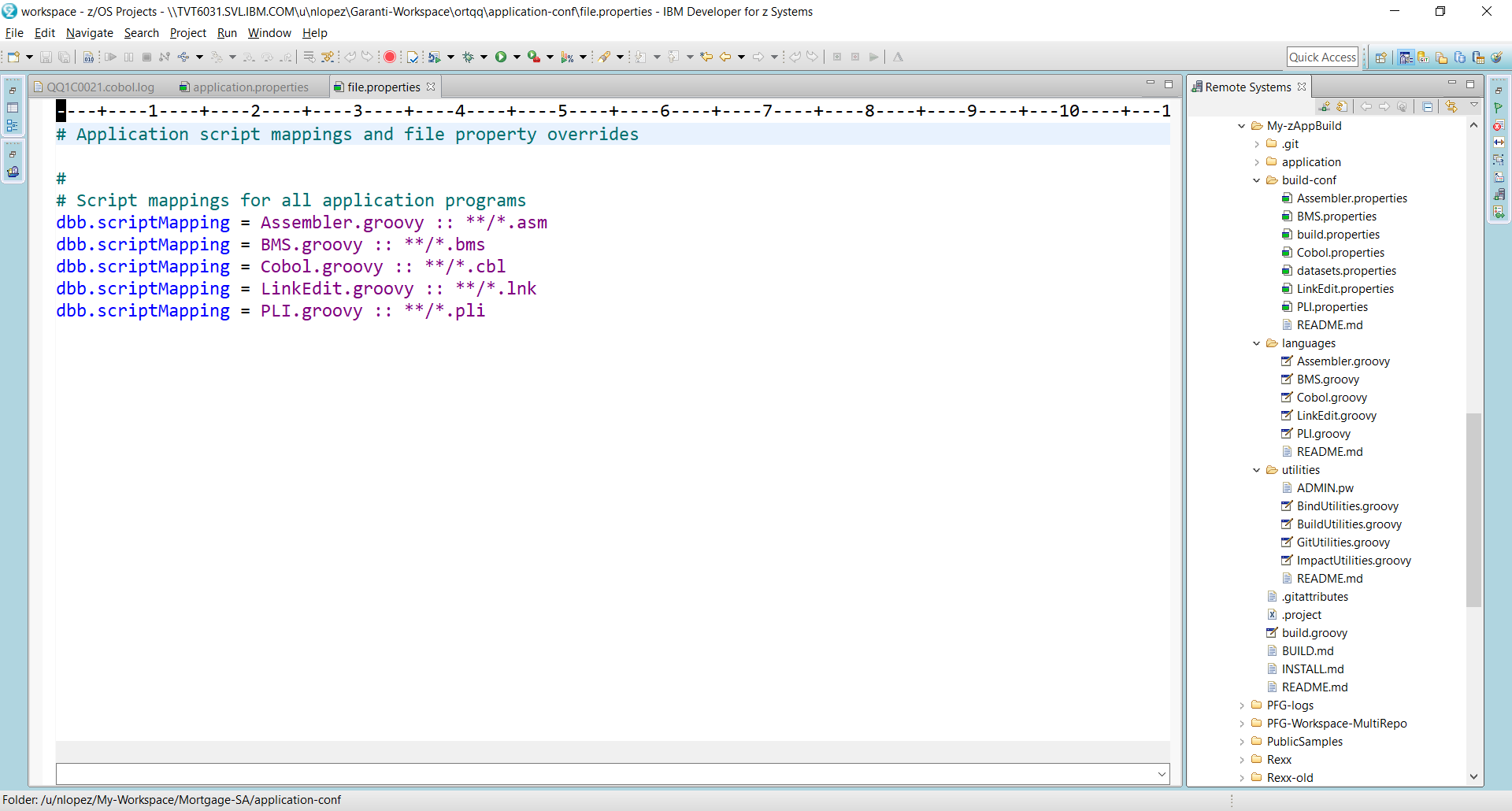
## zAppBuild Intro

zAppBuild is a generic mainframe application build engine to serve as a foundation that customers can extend and customize to meet their DevOps needs. To better understand this framework, think of it as having a global and local scope. The global level consists of a set of company-wide properties and scripts common to all builds and maintained by a central administrative team. The local level allows you to customize properties to meet the specific needs of any application and is maintained by development teams.

The overall design and flow of the build process is illustrated below. There are configuration files that define properties and a set of groovy scripts driven by the main process build.groovy.



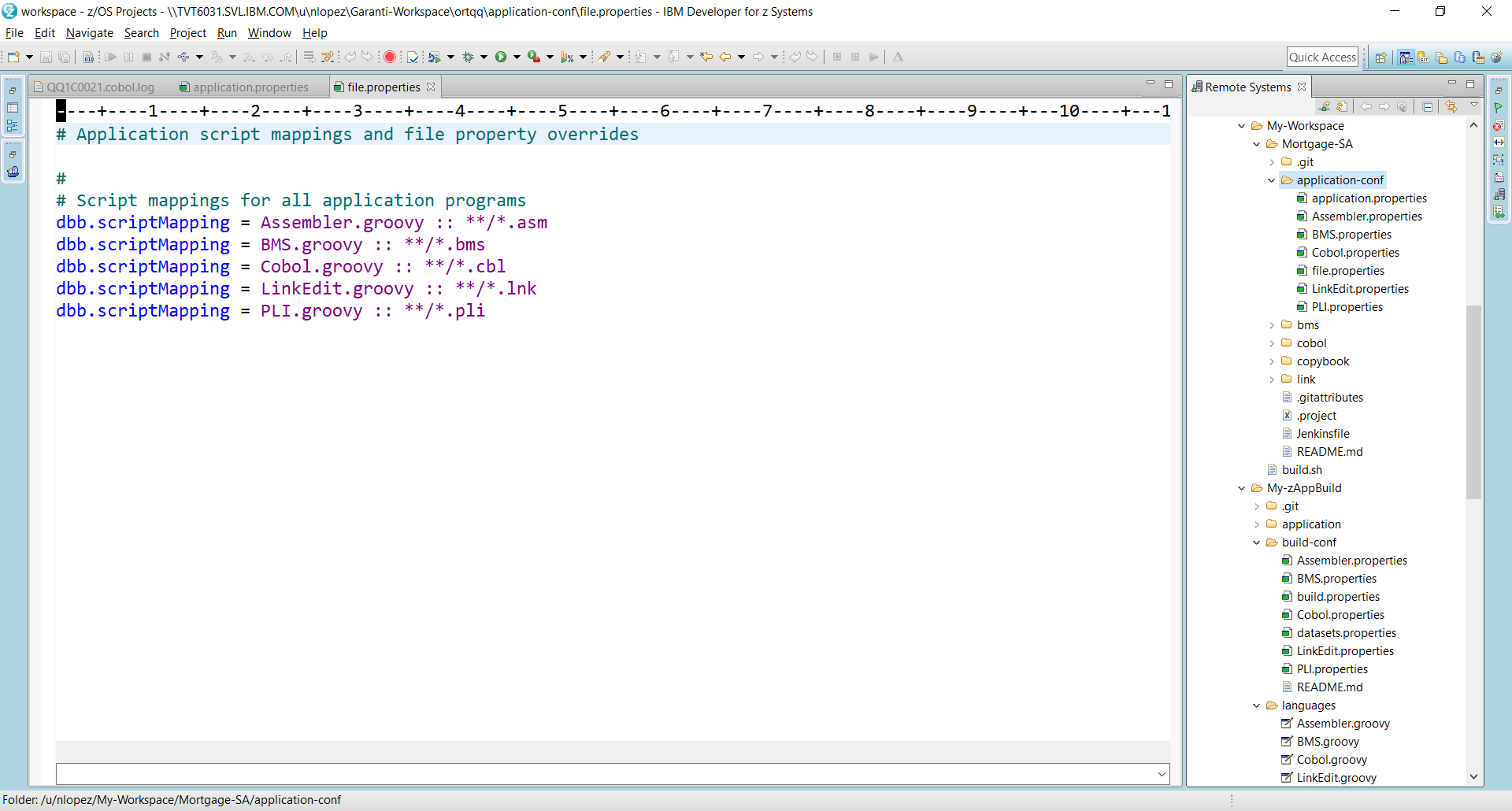


zAppBuild install files are part of the DBB Server installation or can be downloaded/cloned from the link above into a USS folder. Protect zAppBuild from public write access but allow DevOps team members read/write.   
  
zAppBuild is split into two folders. One called “**/zAppBuild**” which is the location of the global items and another called “/**application-conf**” to define application properties.

**“/zAppBuild**” contains the following subfolders and files (in this example the zAppBuild folder is prefixed with “MY-“ but can be called anything:

* **“/application/application-conf”** is a template to be copied and used for your application build. It is not used at the global level only at the local application level (more on that below)

* **“/build-conf**”
  + “**build-properties**” defines the dbb-url, password and other initialization properties. Make sure to update the DBB URL and password information. When using a password file, ensure it is tagged as ISO88591-1 (ascii).
  + **“dataset.properties”** describes system datasets like the PDS name of the Cobol compiler. You must update this with your site’s compiler dataset names.
  + Several language specific property files that define the compiler or link-editor/binder program names, system libraries and general system level properties for Cobol, Assembler …
* **“/languages”** the location of groovy scripts used to build a program. For example, Cobol.groovy is called by build.groovy to compile Cobol source code. These scripts are mapped by file extension in application-conf /file.property.
* **“/utilities”** contains helper scripts used by build.groovy and other scripts.
* **build.groovy** is the main build script. Invoke it to initialize a build environment using global and local properties and to call the appropriate language script based on a source file’s extension.

**/application-conf** is normally part of an application’s source folder structure as shown in this example. The sample application here is called “Mortgage-SA” which is also its repo name. Application folder(s) are contained under a workspace folder. You can have more than one application folder but only one application-conf for the workspace. The application and workspace names should follow your normal naming conventions.

Under “/application-conf” you’ll find:

* **“application.properties”** to describe items like source directory, git branch, copybook lookup rules
* **“file.properties”** maps a source files extension to a groovy build script in zAppBuild/languages
* Various language properties that are applicable to this application.

In this example, it is assumed the My-Workspace was pre-defined and a clone of “Mortgage-SA” was performed under it. Note that the repo has a .git, .gitattributes and a .project file which is used to define the application within IDz’s zOS perspective.

## Additional References

For complete DBB 1.0.6[[1]](#footnote-1) documentation <https://www.ibm.com/support/knowledgecenter/SS6T76_1.0.6/welcome.html>.

To learn basic groovy DBB scripting <https://www.ibm.com/support/knowledgecenter/SS6T76_1.0.6/pg_intro.html>

For an overview of the DBB API’s see <https://www.ibm.com/support/knowledgecenter/SS6T76_1.0.6/zoscommands.html>

For details on the Build Report see <https://www.ibm.com/support/knowledgecenter/SS6T76_1.0.6/buildreport.html>

## Testing and Verification

For testing and verification, it may be best to create a USS shell script to invoke build.groovy where you can pass the required arguments from a command line. Once the code is stable, you can configure IDz and Jenkins to invoke the same zAppBuild configuration.

groovyz $HOME/My-zAppBuild/build.groovy **-w** $HOME/My-Workspace **-a** Mortgage-SA

**-o** $HOME/My-logs **-h** NLOPEZ.DBB.SANDBOX **--fullBuild**

Below is a sample CLI invocation of a zAppBuild run (all one line):

**zAppBuild Arguments:**

* **groovyz** is the DBB initialization script found in the install directory of DBB (resolved using path environment variable)
* **$HOME/My-zAppBuild/build.groovy** – the main zAppBuild driver script
* **-w** the **w**orkspace directory for all related applications
* **-a** an **a**pplication subfolder (relative to -w) where **application-conf** and your application source are defined.
* **-o** directory for the sysprint logs (relative to home)
* **-h** the HLQ for MVS working PDS’s used by DBB
* **--fullBuild** is one of DBB’s build modes
* **-v** is an option to enable verbose logging (not shown in the above example)

## .profile for DBB CLI

For command line processing add these environmental variables to your USS .profile.

export \_BPX\_SHAREAS=NO

export JAVA\_HOME=/usr/lpp/java/J8.0\_64

export DBB\_HOME=/var/dbb

export DBB\_CONF=/var/dbb/conf

export GROOVY\_HOME=$DBB\_HOME/groovy-2.4.12

export PATH=$JAVA\_HOME/bin:$GROOVY\_HOME/bin:$DBB\_HOME/bin:$GIT\_Path:$PATH

. /var/dbb/conf/gitenv.sh

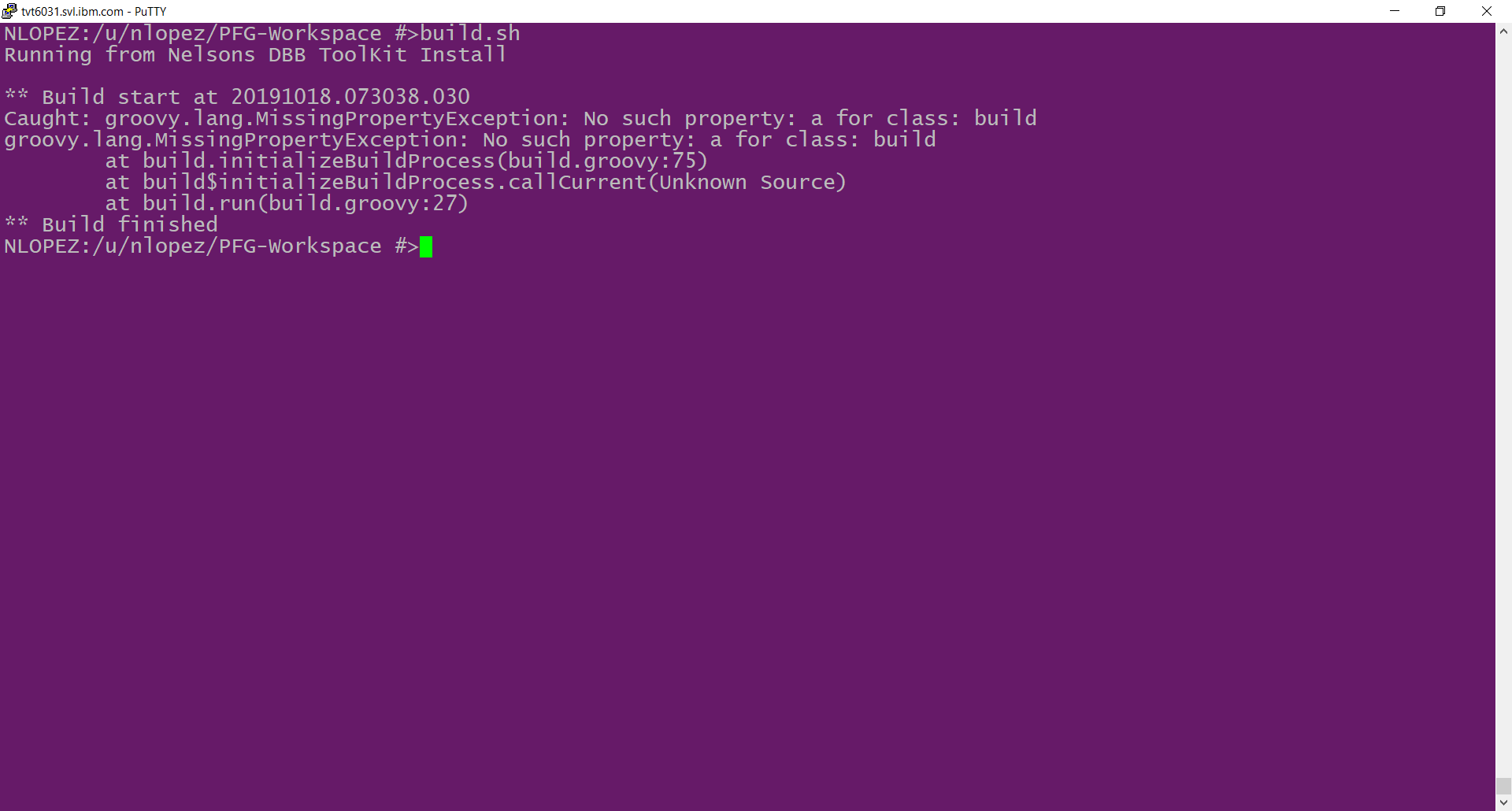
# use Rockets bash

exec /var/rocket/bin/bash

## Debugging tips

The most common cause of runtime errors are typo’s in the property fields (look out for upper/lower case characters).

The most common types of groovy scripting error will fall into these board categories:

1. Script errors- These will produce a standard java trace with the line number of the groovy script in question. Start with the topmost line. In this example, line 75 of build.groovy threw an error.
2. Groovy language related errors – see the groovy programming reference at …

<http://groovy-lang.org/single-page-documentation.html>

1. DBB API errors (prefixed as ‘com.dbb.ibm…’) – see the DBB Javadoc <https://www.ibm.com/support/knowledgecenter/SS6T76_1.0.6/javadoc/index.html> (check your installed version)
2. MVS system errors:
   1. View the MVS system log for SAF(RACF) access issues
   2. SVC 99 Error Codes -<https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.ieaa800/erc.htm#erc__mjfig7>

<https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.bpxb600/rda.htm>

**To help diagnose errors**

* Add groovy println to your script as an aid in tracing issues.
* Search our Git Hub site for known issues or open a new issue at <https://github.com/IBM/dbb>
* Use option -v when invoking build.groovy to enable verbose logging and diagnostics
* Enable log4j -see the DBB installation guide for more details <https://www.ibm.com/support/knowledgecenter/SS6T76_1.0.8/setup_dbb_toolkit.html>

## zAppBuild Complete CLI Arguments

**Standard args**

* -w Absolute path to workspace (root) directory containing all required source directories
* -a Application directory name (relative to workspace)
* -o Absolute path to the build output root directory
* -h High level qualifier for partition data sets

**Build options**

* --propFiles Commas separated list of additional property files to load. Absolute paths or relative to workspace.
* --logEncoding Encoding of output logs. Default is EBCDIC'
* --fullBuild Flag indicating to build all programs for application
* --impactBuild Flag indicating to build only programs impacted by changed files since last successful build.
* --scanOnly Flag indicating to only scan files for application
* --reset Deletes the dependency collections and build result group from the DBB repository
* --verbose Flag to turn on script trace

**WebApp properties - overrides properties in build.properties**

* -url DBB repository URL
* -id DBB repository user id
* -pw DBB repository password
* --pwFile Absolute or relative (from workspace) path to file containing DBB password

**IDz/ZOD User build options**

* --userBuild Flag indicating running a user build
* --errPrefix Unique id used for IDz error message datasets
* --sourceDir Absolute path to workspace (root) directory containing all required source directories for user build
* --workDir Absolute path to the build output root directory for user build
* --team Team build hlq for user build syslib concatenations
* --workspace Absolute path to workspace (root) directory containing all required source directories
* -a Application directory name (relative to workspace)
* -o Absolute path to the build output root directory
* -h High level qualifier for partition data sets

1. Select documentation that matches you installed version. The toolkit version can be found in your installation path ?/dbb/bin/version.properties [↑](#footnote-ref-1)