GitComponentVersion Documentation

Release 0.0.1

Kevin Johnson

Apr 12, 2017

Contents:

This documentation describes a tool that does not yet exist!

GitComponentVersion is a tool to help you achieve *Semantic Versioning* for multiple projects in a single repository.

CHAPTER 1

Installation

GitComponentVersion is available on both nuget.org and Chocolatey

Nuget

Available on nuget.org under `GitComponentVersion.CommandLine`

```
nuget install GitComponentVersion.CommandLine
```

Chocolatey

Available on Chocolatey under `GitComponentVersion.Portable`

```
choco install GitComponentVersion.Portable
```

CHAPTER 2

How it Works

GitComponentVersion uses a combination of a `GitComponentVersion.json` file at the root of your repository, and the git history to determine the version for multiple components within a single repository.

A Short Example

If you have never configured your repository for use with GitComponentVersion you will want to run the following command at root of your repository

```
> gcv init
```

This will assist you to set up your initial GitComponentVersion.json file.

The following `GitComponentVersion.json` file configures a repository with 2 different components (shipping, and billing). Each of these components has 2 directories, for a total of 4 nuget assemblies that each produce their own nuget package. There are no history elements in the config file, meaning that there has never been a release of either of these components.

```
{
    "components": [
        {
            "name": "shipping",
            "next": "1.0.0",
            "tag": "alpha",
            "dirs": [
                "shipping.core",
                "shipping.calculations"
            ]
        },
        {
            "name": "billing",
            "next": "1.0.0",
            "tag": "beta",
            "dirs": [
```

```
                "billing.core",
                "billing.useraccounts"
            ]
        }
    ]
}
```

The following command will display the `NuGetVersion` for each component

```
> gcv -v NuGetVersion
[
    {
        "Name": "shipping",
        "NuGetVersion":"1.0.0-alpha0169",
    },
    {
        "Name": "billing"
        "NuGetVersion":"1.0.0-beta0077",
    }
]
```

This means that there have been 169 commits that modified the shipping component, and 77 that have modified the billing component.

At this point we are ready to make a release of all the components in this repository. Often this means creating a release branch and merging that into the master branch. However, it is not important to GitComponentVersion what workflow you use. From your repository, on the commit you intend to tag as a release, run the following command.

```
> gcv release
```

The following things just happened.

1. Each changed component now has a new tag in the format `{name}_v{next}` In other words, the current commit has now been tagged with `shipping_v1.0.0`, and `billing_v1.0.0`
2. A new release element has been added into each components history section
3. The `next` variable has been bumped to the next minor version for each released component.
4. The `tag` variable has been set to `alpha`

The config file now looks like this:

```
{
    "components": [
        {
            "name": "shipping",
            "next": "1.1.0",
            "tag": "alpha",
            "dirs": [
                "shipping.core",
                "shipping.calculations"
            ],
            "history": [
                {
                    "version": "1.0.0",
                    "sha": "bb2febab25fa1b0312bc61af0116e2de99fa0d5f"
                }
            ]
        },
```

```
        {
            "name": "billing",
            "next": "1.1.0",
            "tag": "alpha",
            "dirs": [
                "billing.core",
                "billing.useraccounts"
            ],
            "history": [
                {
                    "version": "1.0.0",
                    "sha": "4fe97191cc24cfb1f19e3880b3cffa87d10051c7"
                }
            ]
        }
    ]
}
```

It is recommended that the change to GitComponentVersion.json should be committed, and the changes should be pushed to the remote, as well as pushing the new tags

```
> git add GitComponentVersion.json
> git commit -m "Updating the GitComponentVersion.json with new release information"
> git push
> git push --tags
```

At this point your release is complete. You now need to make sure that your development branch is up to date. If you develop against the master branch you are probably already done. However, if you use a master and develop branch you should merge this change into develop so that new development will begin to version with the new version.

CHAPTER 3

Usage

```
usage: gcv [<args>]

verbs:
init                          Sets up the initial GitComponentVersion.json file
release                       adds a record to the history section for a component,
                              tags the repostiory with {name}_v{next},
                              and bumps the [next] version to the next minor version.
update                        Will recursively search for all 'AssemblyInfo.cs' files␣
↪in the component
                              and update them with the correct version number.

arguments:
-c, --component               A comma separated list of component names.
                              Makes a release entry into the history section for the␣
↪named component.
-v, --variable                Restricts output to the specified variable
-h, --help                    Display this help screen.
```

Examples

Display the specified variable for info for a single component

```
> gcv -c core -v NuGetVersion
3.2.0-alpha0169
```

Display the specified variable for info for all components

```
> gcv -v NuGetVersionV2
[
    {
        "Name": "core",
        "NuGetVersionV2":"3.2.0-alpha0169",
```

```
    },
    {
        "Name": "billing"
        "NuGetVersionV2":"3.1.0-alpha0077",
    }
]
```

Display the version info for a single component

```
> gcv -c core
{
    "Major":3,
    "Minor":2,
    "Patch":0,
    "PreReleaseTag":"alpha.169",
    "PreReleaseTagWithDash":"-alpha.169",
    "PreReleaseLabel":"alpha",
    "PreReleaseNumber":169,
    "BuildMetaData":"",
    "BuildMetaDataPadded":"",
    "FullBuildMetaData":"Branch.develop.Sha.88563159817f8ff73897f47119bdb542ef9121db",
    "MajorMinorPatch":"3.2.0",
    "SemVer":"3.2.0-alpha.169",
    "AssemblySemVer":"3.2.0.0",
    "FullSemVer":"3.2.0-alpha.169",
    "InformationalVersion":"3.2.0-alpha0169",
    "BranchName":"develop",
    "Sha":"88563159817f8ff73897f47119bdb542ef9121db",
    "NuGetVersion":"3.2.0-alpha0169",
    "CommitsSinceVersionSource":169,
    "CommitsSinceVersionSourcePadded":"0169",
    "CommitDate":"2017-04-10"
}
```

Display the version info for all components

```
> gcv
[
    {
        "Name": "core",
        "Major":3,
        "Minor":2,
        "Patch":0,
        "PreReleaseTag":"alpha.169",
        "PreReleaseTagWithDash":"-alpha.169",
        "PreReleaseLabel":"alpha",
        "PreReleaseNumber":169,
        "BuildMetaData":"",
        "BuildMetaDataPadded":"",
        "FullBuildMetaData":"Branch.develop.Sha.
↪88563159817f8ff73897f47119bdb542ef9121db",
        "MajorMinorPatch":"3.2.0",
        "SemVer":"3.2.0-alpha.169",
        "AssemblySemVer":"3.2.0.0",
        "FullSemVer":"3.2.0-alpha.169",
        "InformationalVersion":"3.2.0-alpha0169",
        "BranchName":"develop",
        "Sha":"88563159817f8ff73897f47119bdb542ef9121db",
```

```
        "NuGetVersion":"3.2.0-alpha0169",
        "CommitsSinceVersionSource":169,
        "CommitsSinceVersionSourcePadded":"0169",
        "CommitDate":"2017-04-10"
    },
    {
        "Name": "billing"
        "Major":3,
        "Minor":1,
        "Patch":0,
        "PreReleaseTag":"alpha.77",
        "PreReleaseTagWithDash":"-alpha.77",
        "PreReleaseLabel":"alpha",
        "PreReleaseNumber":77,
        "BuildMetaData":"",
        "BuildMetaDataPadded":"",
        "FullBuildMetaData":"Branch.develop.Sha.
↪378037291d0debe840a7cf917ca7e90a914ad390",
        "MajorMinorPatch":"3.1.0",
        "SemVer":"3.1.0-alpha.77",
        "AssemblySemVer":"3.1.0.0",
        "FullSemVer":"3.1.0-alpha.77",
        "InformationalVersion":"3.1.0-alpha0077",
        "BranchName":"develop",
        "Sha":"378037291d0debe840a7cf917ca7e90a914ad390",
        "NuGetVersion":"3.1.0-alpha0077",
        "CommitsSinceVersionSource":77,
        "CommitsSinceVersionSourcePadded":"0077",
        "CommitDate":"2017-04-07"
    }
]
```

release

When no arguments are supplied it is assumed that all *changed* components should be released. Optionally you can specify which components should be considered for release.

The release verb will tag the git repo with the format `{name}_v{next}` and it will add a record to the history section for a component. Then it will bump the `next` version to the next minor version. If the next version should be a major version bump the appropriate component will need to be modified to reflect the correct next version.

```
usage: gcv release [<args>]

-c, --component            A comma separated list of component names.
                           Makes a release entry into the history section for the␣
↪named component.
-n, --dry-run              Don't actually make a release, just show what components␣
↪would be released.
```

update

Recursively search for all `AssemblyInfo.cs` files in the component and update them with the correct version number.

```
usage: gcv update [<args>]

-c, --component            A comma separated list of component names.
                           Specifies the components that will have their assemblyinfo␣
↪files updated.
```

CHAPTER 4

Configuration

Example Configuration File

```
{
    "components": [
        {
            "name": "calculations",
            "next": "2.0.0",
            "tag": "alpha",
            "dirs": [
                "Calculations",
                "Calculations.Core"
            ],
            "assemblyinfo": [
                "AssemblyInfo.cs",
                "SharedAssemblyInfo.cs"
            ],
            "history": [
                {
                    "version": "1.1.0",
                    "sha": "8c8b84ae9f1c7791b69340f3cdb90025822ba77e",
                    "notes": "blah blah blah"
                },
                {
                    "version": "1.0.0",
                    "sha": "8c8b84ae9f1c7791b69340f3cdb90025822ba77e",
                    "notes": "blah blah blah"
                }
            ]
        },
        {
            "name": "billing",
            "next": "1.0.0",
            "tag": "beta",
```

```
            "dirs": [
                "Billing.Core",
                "Billing.Customer"
            ]
        }
    ]
}
```

CHAPTER 5

Indices and tables

- genindex
- search