Versioning Strategy

Version Number Format: **MAJOR.MINOR.PATCH**

Given a version number MAJOR.MINOR.PATCH, increment the:

- MAJOR version when you make incompatible API changes,
- MINOR version when you add functionality in a backwards-compatible manner, and
- PATCH version when you make backwards-compatible bug fixes.

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

Refer Semantic Versioning

Git-Flow Branching

- master
- develop
- feature/*
- bugfix/*
- release/*

Refer GitFlow

Versioning Tasks

- snapshot : Generates the snapshots
- candidate : Generates the release candidates
- final : Generates the final version

A small versioning story

Once upon a time there was an app called ver-demo. The whole code just stayed in just one master branch and thats it.

Jim Halpert joins the team. Unlike others, he wants to do everything smart. He prints the git-flow diagram and pin it on his desk. And the rest of the story follows..

Sets up

Day 1) First, he created `develop` branch from `master`

```
git checkout -b develop
```

He didn't do anything that day, but simply plugged in the develop branch to daily build with generated snapshots.

```
gradle clean build sonarqube snapshot publish
```

This created version `0.1.0-SNAPSHOT` . (Remember: No tags are created on snapshots) artifact was generated - `ver-demo-0.1.0-SNAPSHOT.jar`

Jim knew every day, this would create artifacts under `0.1.0-SNAPSHOT` folder in the artifactory maven-snapshots repository as `name-version-YYYYMMDD.HHmmSS.#.jar`

```
maven-snapshots
    com/dunder-mifflin/hello
        0.1.0-SNAPSHOT
            ver-demo-0.1.0-20170622.154848-1.jar
            ver-demo-0.1.0-20170622.154848-2.jar
```

Build something

Day 2) Michael asks him to build something instead of chatting at reception desk.

Jim goes to his desk and creates feature branch (`feature/build-smthng`) from `develop`

`git checkout -b feature/build-smthng`

He then runs `gradle clean build` and notice the version number it generated `0.1.0-dev.3-build.smthng.cafb6a9`

> It told him 3 commits in develop from the last release and it had his branch name and last commit# too

He changes `System.out.println("Hello.. Its Dunder Mifflin");` to `System.out.println("Hello.. Dwgiht is the worst!")` in Hello.java;

He runs `gradle clean build` again and noticed the change in version number : `0.1.0-dev.3.uncommitted-build.smthng.cafb6a9`

When he ran the app, it printed "Hello.. Dwgiht is the worst!".

Checked the code in `git add. && git commit -m "greatest" && git push` and ran `gradle clean build` command again just for fun and saw the version number changed to `0.1.0-dev.4-build.smthng.d40eaf0`

> 4 commits in develop in this SNAPSHOT version; the last commit is Jim's

It was time to merge his feature to develop. So he did..

```
git checkout develop
git merge feature/build-smthng
```

... and left for the day.

Daily build ran on develop and created a new SNAPSHOT version with Jim's changes

```
maven-snapshots
    com/dunder-mifflin/hello
        0.1.0-SNAPSHOT
            ver-demo-0.1.0-20170622.154848-1.jar
            ver-demo-0.1.0-20170622.154848-2.jar
            ver-demo-0.1.0-20170622.160800-3.jar  <== Jim's changes here
```

Ready for Release (Candidate)

Day 3) Jim wants to push his change to production. So he goes on and creates a release branch.

Jim creates release branch (`release/0.x`)

```
git checkout -b release/0.x
```

He is still not sure if this is the final release version. So he creates a release-candidate (RC)

```
gradle candidate publish
```

This created version `0.1.0-rc.1`

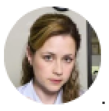
```
<major>.<minor>.<patch>-rc.#
[# is the number of release candidates for this version produced so far]
```

and jar got uploaded to maven-releases repo in artifactory (publish)

```
maven-releases
    com/dunder-mifflin/hello
        0.1.0-rc.1
            ver-demo-0.1.0-rc.1.jar
```

He notices that the git repository has now a tag created (`v0.1.0-rc.1`)

Oops..Bugs

He is all set to push it to prod but wants to show to Pam once .

> "OOps" says Pam.. "Jim, Dwight's name is spelled wrong".

Jim is frustrated but he is determined to fix it and push the release today itself.

Jim creates a bugfix branch from `release/0.x` -> `bugfix/j-123`

```
git checkout -b bugfix/crap-spell
```

He corrects "Dwgiht" to "Dwight" in Hello.java

Builds it `gradle clean build`

version# - `0.1.0-rc.1.dev.0.uncommitted-d40eaf0`

He commits the changes and runs the build:

```
git add && git commit -m "spell fix" && git push && gradle clean build`
```

This time the version# is `0.1.0-rc.1.dev.1-20aadcb`

Merge the fix back

Jim merges `bugfix/crap-spell` branch to `release/0.x`

```
  git checkout release/0.x
  git merge bugfix/crap-spell
```

He creates a new release candidate

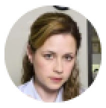
```
gradle candidate publish
```

new RC version `0.1.0-rc.2` got generated and jar got uploaded to maven-releases repo in artifactory (publish)

```
  maven-releases
      com/dunder-mifflin/hello
          0.1.0-rc.1
              ver-demo-0.1.0-rc.1.jar
          0.1.0-rc.2
              ver-demo-0.1.0-rc.2.jar  <== Release candidate 2 (fix for spell error)
```

He notices that the git repository has now a tag created (`v0.1.0-rc.2`)

> "Pam, check it out now.."

Pam verifies and gives him thumbs up. "Looking good Jim" 

Released (finally)

Jim makes the release final

```
gradle final publish
```

That creates the version `0.1.0` and uploads the jar to artifactory

```
  maven-releases
      com/dunder-mifflin/hello
          0.1.0
              ver-demo-0.1.0.jar          <=== RELEASE JAR
          0.1.0-rc.1
              ver-demo-0.1.0-rc.1.jar
          0.1.0-rc.2
              ver-demo-0.1.0-rc.2.jar
```

He notices that the git repository has now a tag created (`v0.1.0`)

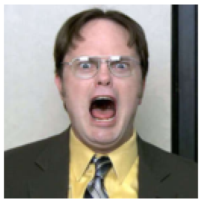
He proceeds with the deployment and Dundler mifflin Hello app now says "Hello...Dwight is the worst!"

Merge

As a responsible guy, Jim syncs up the changes to both master and develop branches

```
git checkout master
git merge release/0.x
git checkout develop
git merge release/0.x
```

Next day, Dwight was in for a major surpise when he started Hello app.

 *"Hello...Dwight is the worst!"*

Now, Dwight wants to prank..

Dwight decides to change "Dwight is the worst" to "_____" (a real solid prank. shhh..)

He creates a new feature branch

```
git checkout -b feature/kill-jim
```

Does a build `gradle clean build`

version # shows up as `0.2.0-dev.0-develop.20aadcb`

He notices the version number has been automatically upgraded to **0.2.0 as 0.1.0 has already been released. Anything from now on is > 0.1.0**

The story continues.. (but hope you got the point)

Keep in mind

- snapshot : Generates the snapshots but doesn't create tag in repository
- candidate : Generates the release candidates and creates tag in repository (v$ver-rc.#)
- final : Generates the final version and creates tag in repository (v$ver)

Typically,

- `snapshot` is done on `develop` branch
- `candidate` and `final` are done on `release` branches
  - can be done on `bugfix` branch for hotfixes

- on release/1.x branch, after every release, the version number increases as v1.*x+1*.0
- on release/2.1.x branch, after every release, the version number increases as v2.1.*x+1*
- major, minor or patch numbers can be bumped up manually also
  - (gradle plugin does it this way `gradle <snapshot|candidate|release> -Prelease.scope=<major|minor|patch>`)