Practical Series

Version Control:  
A numbering mechanism

By Michael Gledhill

|  |
| --- |
| Revision: R01.00—18 May 2017 |

|  |  |  |  |
| --- | --- | --- | --- |
| Copyright 2017 | Practical Series Publications | | |
| All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except under the terms of the Copyright Designs and Patents Act 1988 without the permission in writing of the publisher. | | | |
| This publication was set using Equity, Concourse and Triplicate fonts. | | | |
|  | | | |
| Published By: | Practical Series Publications Chester, Great Britain  Email: mg@practicalseries.com | |  |
|  |  | | |
| Draft Edition | 2017 | Document No.: PS1002-5-2121-091 | |
|  | Based on Template: T0-1101-007 R01.01 Formal Publication (Equity).Dotm | | |
|  |  | | |
| Revision History |  | | |
| R01.00 | 18 May 2017 | First release | |
|  |  | | |
|  |  | | |
|  |  | | |

Contents

A A revision numbering mechanism 5

A.1 Workflow arrangements 8

A.2 Master branch revision states 9

A.3 Development branch revision states 10

A.4 Parallel development branches 13

A.5 Nested development branches 14

A.6 A note on numbering 15

Table of Figures

**Figure A.1 master branch workflow 8**

**Figure A.2 development branch workflow 12**

**Figure A.3 Parallel development branches 13**

**Figure A.4 Merged parallel development branches 13**

**Figure A.5 Nested development branches 14**

Table of Tables

**Table A.1 Major revision point 9**

—NON PRINTING COMMENT—

This page is completely blank (no page numbers).

It is used to make the section text start on an ODD numbered page; giving a logical start to the section body text for double sided documents. To do this, the Chapter Page (previous page) and this one have their own section within the document (allowing the first odd footer to be blank).

Copy point 2

|  |
| --- |
| A revision numbering mechanism |
| A |
| A revision numbering mechanism |
| A revision numbering strategy for a website development project under the Git and GitHub version control systems. |

—NON PRINTING COMMENT—

This page follows a chapter heading and is completely blank (no page numbers).

It is used to make the section text start on an ODD numbered page; giving a logical start to the section body text for double sided documents. To do this, the Chapter Page (previous page) and this one have their own section within the document (allowing the first odd footer to be blank).

This section describes a revision numbering strategy for a website development project under the Git and GitHub version control systems.

Git and GitHub use commit numbers derived from the checksum of files being added to the repository. These appear at best to be seemingly random seven digit hexadecimal numbers. They do not represent a meaningful number that is useful for team members trying to identify a revision path.

Git and GitHub allow any commit point to have an associated tag, this is entirely at the discretion of the user and (other than the requirement to be unique) can be anything at all.

This allows each commit point to be tagged with a more meaningful (semantic) version number. Something that makes sense to humans.

The following is a proposed mechanism for just such a tagging arrangement.

|  |  |
| --- | --- |
| A.1 | Workflow arrangements |
| * 1. Workflow arrangements |

The workflow consists of a single main branch, the master branch.

The master branch (after some initial development work to establish it) will only contain either finished publishable work or a completed and released website.

Publishable work is a section of the website that is complete in itself—it does not indicate that the whole website is finished, just that the section in question is complete, tested and deployable.

A website release indicates that the whole website is finished. Release R01 will be active when the entire website and all its pages is published for the first time.

Development work can take place at any time and will always take place on a separate branch. Development branches always spur from some published or released point on the master branch.

A development branch must have a very restricted scope. E.g. a single section of the website (or just a page).

Generally, a development branch will contain all the things associated with that section: html, js and css files as well as associated images and data.

|  |
| --- |
|  |
| Figure A.1 master branch workflow |

|  |  |
| --- | --- |
| A.2 | Master branch revision states |
| * 1. Master branch revision states |

The project progresses through various different states along the master branch.

These states are given a letter that represents the state of a particular commit point.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | State | Example | Meaning |  |  |
|  | D | D02 | Development | The site is entirely under development and no part of it has been published |  |
|  | P | P04 | Published (partial) | Partial publication—certain pages of the website are finished and have been published |  |
|  | R | R01 | Released | Released—all pages are complete and the whole site is finished |  |
|  | Table A.1 Major revision point | | | |  |

Master branch states have the following format

|  |  |  |
| --- | --- | --- |
|  | SNN |  |
|  |  |  |

Where S is the state letter (Table A.1)

D — Development

P — Published (partial)

R — Released

NN is a number; this starts at 01 for each particular state and is incremented by one for each subsequent issue.

E.g. D01 → D02 → D03 → P01 → P02 → P03 → R01 → R02 &c.

|  |  |
| --- | --- |
| A.3 | Development branch revision states |
| * 1. Development branch revision states |

Development can take place along the master branch until some suitable (publishable) point is reached.

At this point the site has been published, the site might not be complete (with the Practical Series Web Development site I tended to write a section, test it and then publish that section), but no further development work can take place on the master branch.

This leads to the website being developed and published in stages; however, any code on the master branch at a published or released commit point must be finished, fully tested, working and deployable.

Development work always takes place on a separate development branch. Each development branch is taken from the latest published or released version of the software on the master branch.

The name given to a development branch is:

|  |  |  |
| --- | --- | --- |
|  | d-XX-name |  |

Where d- indicates the branch is for development (all my branches are development branches and always start d-)

XX refers to the number of the web page or section being developed. For example, the PS Web Development is split into sections and each section is given a two digit number. It starts like this:

01-introduction

02-fairfax

03-grid

In Figure A.1 the first development branch is called d-99-typicals. This is a page that contains an example of all the possible sections, headings, tables, images and advanced functions such as code fragments, formulae and lightbox imagery.

It is a good starting point for any other page that may be required and it makes sense to develop this page first.

All the development work takes place on the development branch. There will be multiple commits made on this branch; these will mostly be incremental builds (an incremental build is just a point at which the work was committed, the work wasn’t finished; I may just have committed the code at that point because it was the end of the day).

Each development commit on a development branch is tagged, and the tag has the format:

|  |  |  |
| --- | --- | --- |
|  | d-SNN.XX.YY |  |

Where SNN is the state of the commit point on the master branch when the development branch was created (i.e. the parent commit point to which the branch is linked).

XX refers to the number of the web page or section being developed (this is the same as the XX in the branch name, see above).

YY is a number starting at 01 and is incremented by one for each commit along the branch.

Once the code has been developed, it progresses to a test stage; this takes place along the same branch. The commit tag changes however, it now becomes:

|  |  |  |
| --- | --- | --- |
|  | t-SNN.XX.YY |  |

It is now preceded by a t- instead of a d-. The number YY continues at the next number following the last development commit when the test stage starts. This allows tested code to drop back into development. The following sequence illustrates this:

|  |
| --- |
| d-P01.02.01 → **t-P01.02.03** → d-P01.02.04 → **t-P01.02.05** → s-P01.02.P02 |

Finally, at the end of testing, the code is finished and is ready to be merged back into the master branch. At this point, a new commit point is made on the branch. This is referred to a Staged commit. It again has a tag and this time the tag format is:

|  |  |  |
| --- | --- | --- |
|  | s-SNN.XX.SNN+k |  |

The s- indicates a staged commit. SNN.XX is exactly the same as each other commit on the branch

SNN+k indicates the new merge commit point on the master branch that will be created when the development branch is merged onto the master branch.

Normally, this will just be a the SNN value plus one (it may be more if other branches are involved, it will always be the next publication or release commit number on the master branch, see § A.4).

In Figure A.2, the development branch d-99-typiclas split from the master branch at commit point P01 (SNN is P01). After the development process, the code is ready to merge back onto the master branch. The next publication commit point on the master branch will be P02 (NN goes up by 1). So the last staged commit on the development branch will be s-P01.99.**P02**.

|  |
| --- |
|  |
| Figure A.2 development branch workflow |

The final commit on the development branch, the staged commit (s-P01.99.P02 in this case) is done to allow the software versions of each file to be changed to match the forthcoming master branch commit (P02 in this case). The code itself shouldn’t change, just its revision status.

There is only ever one staged commit on a development branch.

|  |  |
| --- | --- |
| A.4 | Parallel development branches |
| * 1. Parallel development branches |

It is perfectly possible to have two simultaneous development branches

|  |
| --- |
|  |
| Figure A.3 Parallel development branches |

Note: With parallel branches, it does not matter what order the branches are made and what order they collapse, here d-01 is created first, but d-02 is merged back into the master before it.

This type of arrangement can appear slightly confusing when all the branches are merged back onto the master branch, it looks like this (I’ve coloured the tags with the appropriate branch colour):

|  |
| --- |
|  |
| Figure A.4 Merged parallel development branches |

It makes sense if you think about it; commits are listed in order of the time they were applied.

|  |  |
| --- | --- |
| A.5 | Nested development branches |
| * 1. Nested development branches |

It is possible to have a development branch from another development branch (referred to as nesting):

|  |
| --- |
|  |
| Figure A.5 Nested development branches |

The second development branch is called da-<as parent branch>. The third would be called db-<as parent branch>, da, db, dc &c.

Each commit on the nested branch has the format:

|  |  |  |
| --- | --- | --- |
|  | d/t/sL-SNN.XX.ZZ |  |
|  |  |  |

Where L is the branch identifier (a, b, c, &c).

SNN-XX is identical to the parent branch (master branch commit and web page or section number).

ZZ is a number starting at 01 and is incremented by one for each commit along the nested branch.

Nested branches can have test revisions (ta, tb, tc, &c.) and a staged commit (sa, sb, sc &c.).

The staged commit reflects the merged commit number on the parent branch. There is only one staged commit per nested branch.

|  |  |
| --- | --- |
| A.6 | A note on numbering |
| * 1. A note on numbering |

All the numbering I’ve shown in this section has been two digits d-P01.03.05. This clearly limits any revision, page or section number to a maximum value of 99.

This needn’t be the case; it is perfectly possible to use three digit numbers d‑P001.003.005, indeed this could be selectively adapted d-P01.0103.005, you can do whatever you like.

I think the two digit version is enough for me. If I’ve got more than 99 commits on a development branch I’m probably doing it wrong.

I can see the advantage in having a four digit middle number d-P01.0103.05, it would allow me to resolve to individual web pages; however, I nearly always develop a full section at a time. I write it in Word and then encode it in html.

I suggest you start with the two digit version and see how you get on.

—NON PRINTING COMMENT—

This page follows a chapter heading and is completely blank (no page numbers).

It is used to make the section text start on an ODD numbered page; giving a logical start to the section body text for double sided documents. To do this, the Chapter Page (previous page) and this one have their own section within the document (allowing the first odd footer to be blank).