VIRTUALIZED LEARNING

SVN Guide

Prepared By:

Kaylee Cox	George Matthew
Remy Mezebish	Kyle Nguyen
Neehar Peri	Daniel Rowe
Charlie Shin	Kaitlyn Won

Distributed May, 2021

Prepared For:

Dr. Rodolphe Gentili -

Associate Professor of Kinesiology at the University of Maryland, College Park

Dr. Garrett Katz -

Assistant Professor of Electrical Engineering and Computer Science at Syracuse University

This document discusses our SVN procedures/set-up and how to update the virtual machine/server with the most recent SVN tag.

First and foremost, read this article as it discusses basic SVN concepts such as branching and tagging that we make use of in our SVN repository:

https://betterexplained.com/articles/a-visual-guide-to-version-control/

If you are familiar with Git flow, this should all be very familiar to you. SVN works similarly to Git except that the major difference is that the branches are just another copy of the whole project.

Explanation of Branches

Trunk

This branch is where development code becomes production code. Individual features from the Team Branches are merged together and made working before a tagged release is made. This process will be explained more later.

Team Branches

Teams work individually in these branches, reverse integrating features to trunk and forward integrating updates from trunk. Team branches only interact with each other through the trunk branch. Make sure to communicate with the every team before merging to the trunk.

Tags

Tagged revisions are copies of the trunk that are frozen in time and deployed to the production server for the client or public to use. The code in the Trunk must be **fully tested and verified working** before a tag is made. Make sure to communicate before merging into the trunk to limit interruptions to this process. Search up Semantic Versioning for an explanation on how to name versions.

The Flow

Team/Branch -> Trunk

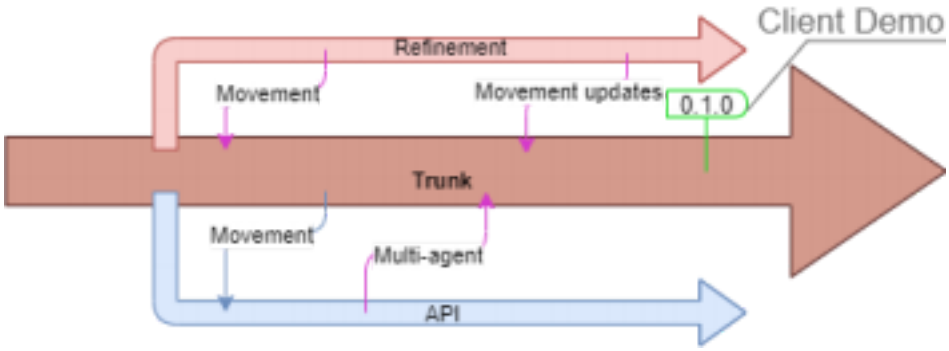
Team members work inside their respective branches. Once they've reached a complete point for a feature, they confirm that tests are passing then notify someone who hasn't worked on their feature to review their work. The reviewer reads over the changes and verifies that feature tests have passed then signs off on it and conducts the merge. Code may require integration with other team's features and can be merged into the trunk as long as it is communicated to every team.

Trunk -> Tag

Code in the trunk is reviewed and tested in preparation for the [tagged] (#tags) release. Someone, preferably from the plug-in team reads over the changes and tests the code against

the trunk. The plug-in team will manually run integration testing. If bugs occur, they are patched by revisions directly to the trunk after review. The testing and patching cycle continues until the tag is ready. After publishing a tag, Team Branches are updated to the latest version by reverse integrating from the trunk.

The diagram below is a high-level view of our SVN repository and how we set up our development structure:



As you can see above, we have set up our development environment to have three branches: refinement, api, and trunk. Feel free to use our development structure or make your own branches based on the needs of your project. Each branch corresponds to our sub-teams. For example, the refinement team will work and make their changes into the refinement branch and the API team will work and make their changes in the API branch.

Merging

In order to merge a branch into another, svn has a merge command. Simply go into the directory/branch that you would like to merge into and enter this command:

```
svn merge [PATH OF BRANCH TO MERGE FROM]
```

If there are merge conflicts, we have found that using the Source Control in Visual Studio Code makes resolving merge conflicts significantly easier.

Read up on this documentation on resolving merge conflicts with Visual Studio Code for more information:

https://code.visualstudio.com/docs/editor/versioncontrol

Once the changes are merged into the trunk, you must commit the changes within the trunk into the SVN repository. In our project, we followed a naming convention for the commits that involved merging. The convention is as follows:

[Merge] Pulling BRANCH into BRANCH through [revision number]

So in order to commit the changes, type this command from the trunk directory:

svn commit -m “[Merge] Pulling BRANCH into BRANCH through [revision number]”

Note: If you have made changes to the unity projects, you should probably build the Unity Projects first and deploy them to the webapp before committing so that you don't have to commit multiple times when merging. You can learn about building unity code from our Building Unity Code document.

Creating a Tag

After the trunk has been merged into and everything has been tested and ready to be deployed to the server, you need to create a tag for this most recent version of the project.

You can either make a new tag yourself in the tags folder and copy over the `vlearn_webapp/` directory within the trunk into that tags folder using svn commands or you can use our tag-creation script that automates the process for tag creation.

Our tag-creation script using the naming convention of npm versioning, more info can be found here:

https://docs.npmjs.com/cli/v7/commands/npm-version

To run the tag creation script, run the command from within the `trunk/vlearn_webapp` directory:

```
./create-tag.sh "[NPM VERSIONING]"
```

For example, if I wanted to create a new tag for a major version of the build, I would use the command:

```
./create-tag.sh "major"
```

Now your tag has been made and is ready to be deployed into the server!

Updating VM/Server to Different Tags

The great thing about tags is that you can easily switch the version of your project that the server is running. If you accidentally made a tag that messes up the server, simply switch back to an older tag!

SVN supports a switch command that can easily switch between tags.

To switch between tags, enter this command from within the `home/vlearn_webapp/` directory in the server:

```
svn switch [URL OF YOUR SVN]/Webapp/tags/[INSERT TAG HERE]/vlearn_webapp/
```

Now your server should be running on the tag that you have specified.

Once again, if there are merge conflicts when switching tags, you can easily resolve them using the Source Control Visual Studio Code extension.

Stop Download of Tags Folder on your Local Machine (Optional)

Because tags are essentially just a copy of your entire project, creating tags can take up space on your local machine and you may want to not download the tags folder on your local machine.

To exclude the tags folder when you use `svn up`, enter this command from the Webapp folder:

```
svn update --set-depth exclude tags
```

To get the tags folder back, enter this command from the Webapp folder:

```
 svn update --set-depth infinity
```