## DevOps

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.



## How DevOps Works

Under a DevOps model, development and operations teams are no longer “siloed.” Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function.

In some DevOps models, quality assurance and security teams may also become more tightly integrated with development and operations and throughout the application lifecycle. When security is the focus of everyone on a DevOps team, this is sometimes referred to as DevSecOps.

These teams use practices to automate processes that historically have been manual and slow. They use a technology stack and tooling which help them operate and evolve applications quickly and reliably. These tools also help engineers independently accomplish tasks (for example, deploying code or provisioning infrastructure) that normally would have required help from other teams, and this further increases a team’s velocity.

## Benefits of DevOps

### Speed

Move at high velocity so you can innovate for customers faster, adapt to changing markets better, and grow more efficient at driving business results. The DevOps model enables your developers and operations teams to achieve these results. For example, [microservices](https://aws.amazon.com/devops/what-is-devops/#microservices) and [continuous delivery](https://aws.amazon.com/devops/continuous-delivery/) let teams take ownership of services and then release updates to them quicker.

### 

### Rapid Delivery

Increase the frequency and pace of releases so you can innovate and improve your product faster. The quicker you can release new features and fix bugs, the faster you can respond to your customers’ needs and build competitive advantage. [Continuous integration](https://aws.amazon.com/devops/continuous-integration/) and [continuous delivery](https://aws.amazon.com/devops/continuous-delivery/) are practices that automate the software release process, from build to deploy.

### Reliability

Ensure the quality of application updates and infrastructure changes so you can reliably deliver at a more rapid pace while maintaining a positive experience for end users. Use practices like [continuous integration](https://aws.amazon.com/devops/continuous-integration/) and [continuous delivery](https://aws.amazon.com/devops/continuous-delivery/) to test that each change is functional and safe. [Monitoring and logging](https://aws.amazon.com/devops/what-is-devops/#monitoring) practices help you stay informed of performance in real-time.

### Scale

Operate and manage your infrastructure and development processes at scale. Automation and consistency help you manage complex or changing systems efficiently and with reduced risk. For example, [infrastructure as code](https://aws.amazon.com/devops/what-is-devops/#iac) helps you manage your development, testing, and production environments in a repeatable and more efficient manner.

### Improved Collaboration

Build more effective teams under a DevOps cultural model, which emphasizes values such as ownership and accountability. Developers and operations teams [collaborate](https://aws.amazon.com/devops/what-is-devops/#communication) closely, share many responsibilities, and combine their workflows. This reduces inefficiencies and saves time (e.g. reduced handover periods between developers and operations, writing code that takes into account the environment in which it is run).

### Security

Move quickly while retaining control and preserving compliance. You can adopt a DevOps model without sacrificing security by using automated compliance policies, fine-grained controls, and configuration management techniques. For example, using infrastructure as code and [policy as code](https://aws.amazon.com/devops/what-is-devops/#policyascode), you can define and then track compliance at scale.

# Jenkins

Welcome to the Jenkins user documentation - for people wanting to use Jenkins’s existing functionality and plugin features.

## If you want to extend the functionality of Jenkins by developing your own Jenkins plugins, please refer to the [Extend Jenkins](https://jenkins.io/doc/developer/) (developer documentation).

## What is Jenkins?

Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

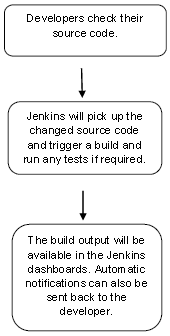
## Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.

## What is Continuous Integration?

Continuous Integration is a development practice that requires developers to integrate code into a shared repository at regular intervals. This concept was meant to remove the problem of finding later occurrence of issues in the build lifecycle. Continuous integration requires the developers to have frequent builds. The common practice is that whenever a code commit occurs, a build should be triggered.

## Why Jenkins?

Jenkins is a software that allows **continuous integration**. Jenkins will be installed on a server where the central build will take place. The following flowchart demonstrates a very simple workflow of how Jenkins works.



Along with Jenkins, sometimes, one might also see the association of **Hudson**. Hudson is a very popular open-source Java-based continuous integration tool developed by Sun Microsystems which was later acquired by oracle.

## System Requirements

|  |  |
| --- | --- |
| JDK | JDK 1.5 or above |
| Memory | 2 GB RAM (recommended) |
| Disk Space | No minimum requirement. Note that since all builds will be stored on the Jenkins machines, it has to be ensured that sufficient disk space is available for build storage. |
| Operating System Version | Jenkins can be installed on Windows, Ubuntu/Debian, Red Hat/Fedora/CentOS, Mac OS X, openSUSE, FReeBSD, OpenBSD, Gentoo. |
| Java Container | The WAR file can be run in any container that supports Servlet 2.4/JSP 2.0 or later.(An example is Tomcat 5). |

## Download Jenkins

The official website for Jenkins is [Jenkins](https://jenkins-ci.org/). If you click the given link, you can get the home page of the Jenkins official website as shown below.



By default, the latest release and the Long-Term support release will be available for download. The past releases are also available for download. Click the Long-Term Support Release tab in the download section.



Click the link “Older but stable version” to download the Jenkins war file.

## Starting Jenkins

Open the command prompt. From the command prompt, browse to the directory where the jenkins.war file is present. Run the following command

D:\>Java –jar Jenkins.war

After the command is run, various tasks will run, one of which is the extraction of the war file which is done by an embedded webserver called winstone.

*D:\>Java –jar Jenkins.war*

*Running from: D:\jenkins.war*

*Webroot: $user.home/ .jenkins*

*Sep 29, 2015 4:10:46 PM winstone.Logger logInternal*

*INFO: Beginning extraction from war file*

Once the processing is complete without major errors, the following line will come in the output of the command prompt.

*INFO: Jenkins is fully up and running*

## Accessing Jenkins

Once Jenkins is up and running, one can access Jenkins from the link − **http://localhost:8080**

This link will bring up the Jenkins dashboard.

# Accessing Jenkins

## JENKINS – TOMCAT SETUP

## Step 1: Verifying Java Installation

To verify Java installation, open the console and execute the following java command.

|  |  |  |
| --- | --- | --- |
| **OS** | **Task** | **Command** |
| Windows | Open command console | \>java –version |
| Linux | Open command terminal | $java –version |

If Java has been installed properly on your system, then you should get one of the following outputs, depending on the platform you are working on.

|  |  |
| --- | --- |
| **OS** | **Output** |
| Windows | Java version "1.7.0\_60"  Java (TM) SE Run Time Environment (build 1.7.0\_60-b19)  Java Hotspot (TM) 64-bit Server VM (build 24.60-b09, mixed mode) |
| Linux | java version "1.7.0\_25"  Open JDK Runtime Environment (rhel-2.3.10.4.el6\_4-x86\_64)  Open JDK 64-Bit Server VM (build 23.7-b01, mixed mode) |

We assume the readers of this tutorial have Java 1.7.0\_60 installed on their system before proceeding for this tutorial.

In case you do not have Java JDK, you can download it from the link [Oracle](http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html)

## Step 2: Verifying Java Home Path To JDK

Set the JAVA\_HOME environment variable to point to the base directory location where Java is installed on your machine. For example,

|  |  |
| --- | --- |
| **OS** | **Output** |
| Windows | Set Environmental variable JAVA\_HOME to C:\ProgramFiles\java\jdk1.7.0\_60 |
| Linux | export JAVA\_HOME=/usr/local/java-current |

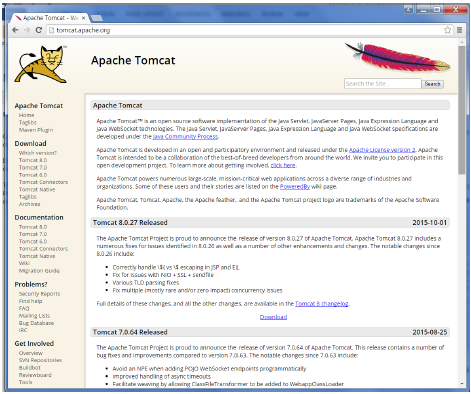
Append the full path of the Java compiler location to the System Path.

|  |  |
| --- | --- |
| **OS** | **Output** |
| Windows | Append the String; C:\Program Files\Java\jdk1.7.0\_60\bin to the end of the system variable PATH. |
| Linux | export PATH=$PATH:$JAVA\_HOME/bin/ |

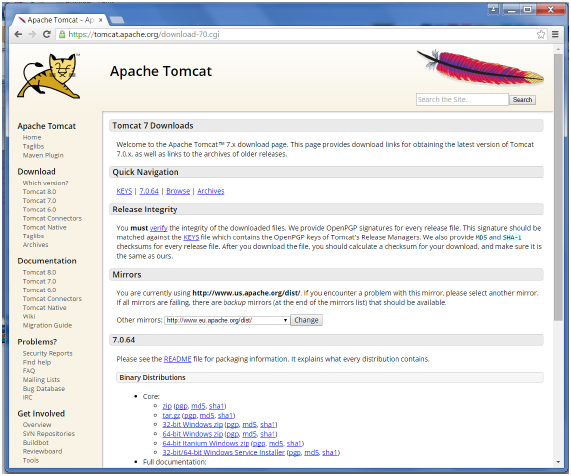
Verify the command java-version from command prompt as explained above.

## Step 3: Download Tomcat

The official website for tomcat is [Tomcat](http://tomcat.apache.org/). If you click the given link, you can get the home page of the tomcat official website as shown below.



Browse to the link <https://tomcat.apache.org/download-70.cgi> to get the download for tomcat.



Go to the ‘Binary Distributions’ section. Download the 32-bit Windows zip file.

Then unzip the contents of the downloaded zip file.

## Step 4: Jenkins and Tomcat Setup

Copy the Jenkis.war file which was downloaded from the previous section and copy it to the webapps folder in the tomcat folder.

Now open the command prompt. From the command prompt, browse to the directory where the tomcat7 folder is location. Browse to the bin directory in this folder and run the start.bat file

E:\Apps\tomcat7\bin>startup.bat

Once the processing is complete without major errors, the following line will come in the output of the command prompt.

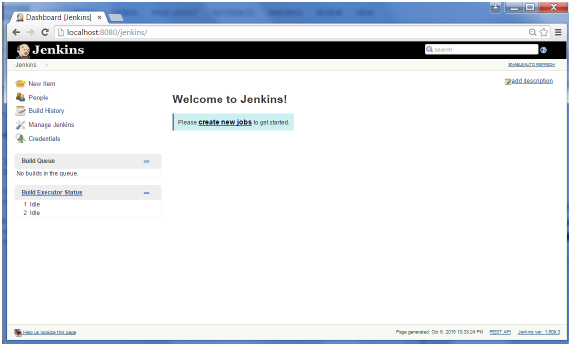
INFO: Server startup in 1302 ms

Open the browser and go to the link − **http://localhost:8080/jenkins**. Jenkins will be up and running on tomcat.

# Jenkins and Tomcat Setup

**JENKINS – GIT SETUP**

For this exercise, you have to ensure that Internet connectivity is present from the machine on which Jenkins is installed. In your Jenkins Dashboard (Home screen), click the Manage Jenkins option on the left hand side.



In the next screen, click the ‘Manage Plugins’ option.



In the next screen, click the Available tab. This tab will give a list of plugins which are available for downloading. In the ‘Filter’ tab type ‘Git plugin’



The list will then be filtered. Check the Git Plugin option and click on the button ‘Install without restart’



The installation will then begin and the screen will be refreshed to show the status of the download.



Once all installations are complete, restart Jenkins by issue the following command in the browser. **http://localhost:8080/jenkins/restart**

After Jenkins is restarted, Git will be available as an option whilst configuring jobs. To verify, click on New Item in the menu options for Jenkins. Then enter a name for a job, in the following case, the name entered is ‘Demo’. Select ‘Freestyle project’ as the item type. Click the Ok button.



In the next screen, if you browse to the Source code management section, you will now see ‘Git’ as an option.



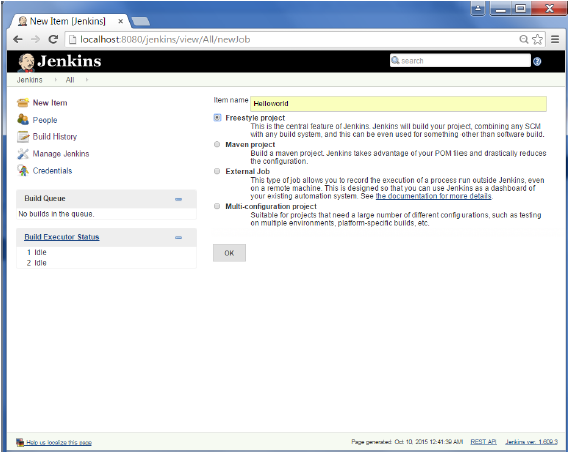
**JENKINS – SETUP BUILD JOBS**

For this exercise, we will create a job in Jenkins which picks up a simple HelloWorld application, builds and runs the java program.

**Step 1** − Go to the Jenkins dashboard and Click on New Item

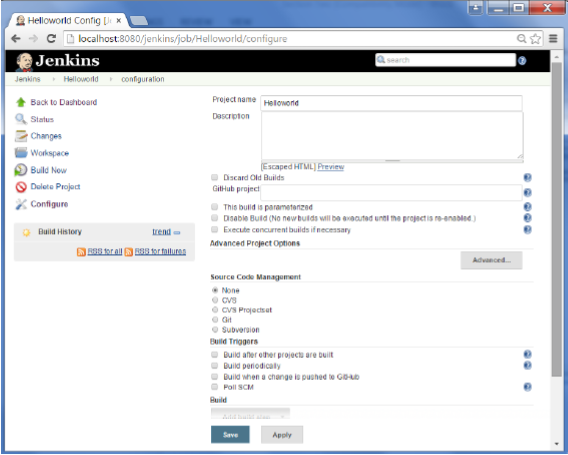


**Step 2** − In the next screen, enter the Item name, in this case we have named it Helloworld. Choose the ‘Freestyle project option’



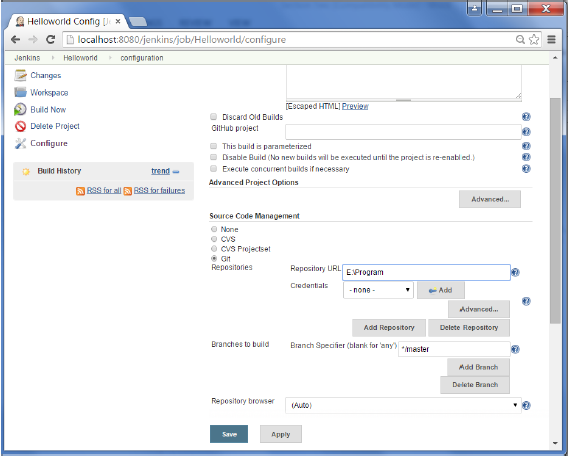
**Step 3** − The following screen will come up in which you can specify the details of the job.

Software and the Internet have transformed the world and its industries, from shopping to entertainment to banking. Software no longer merely supports a business; rather it becomes an integral component of every part of a business. Companies interact with their customers through software delivered as online services or applications and on all sorts of devices. They also use software to increase operational efficiencies by transforming every part of the value chain, such as logistics, communications, and operations. In a similar way that physical goods companies transformed how they design, build, and deliver products using industrial automation throughout the 20th century, companies in today’s world must transform how they build and deliver software.

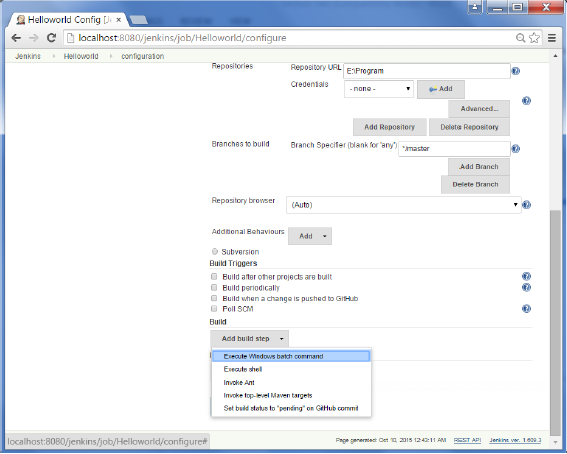


**Step 4** − We need to specify the location of files which need to be built. In this example, we will assume that a local git repository(E:\Program) has been setup which contains a ‘HelloWorld.java’ file. Hence scroll down and click on the Git option and enter the URL of the local git repository.

**Note** − If you repository if hosted on Github, you can also enter the url of that repository here. In addition to this, you would need to click on the Add button for the credentials to add a user name and password to the github repository so that the code can be picked up from the remote repository.



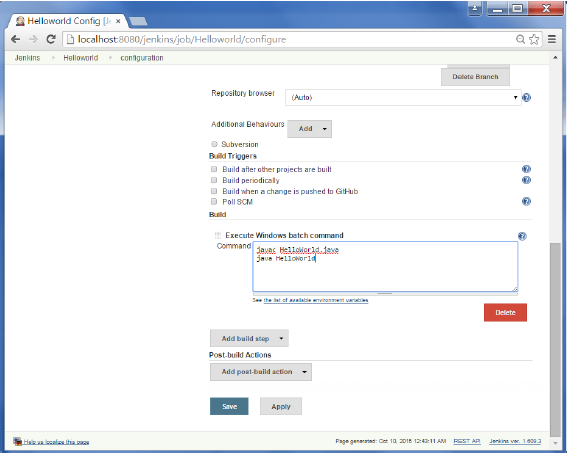
**Step 5** − Now go to the Build section and click on Add build step → Execute Windows batch command.



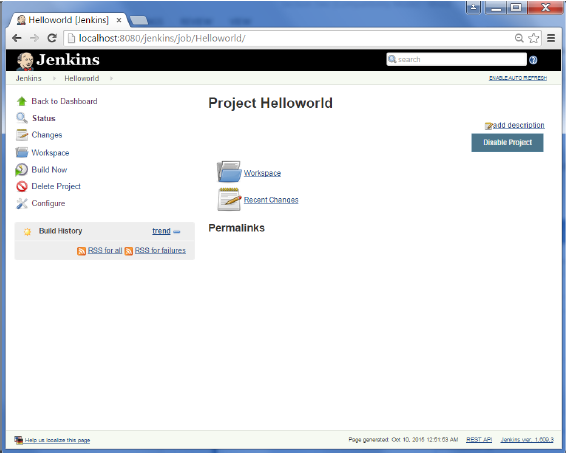
**Step 6** − In the command window, enter the following commands and then click on the Save button.

Javac HelloWorld.java

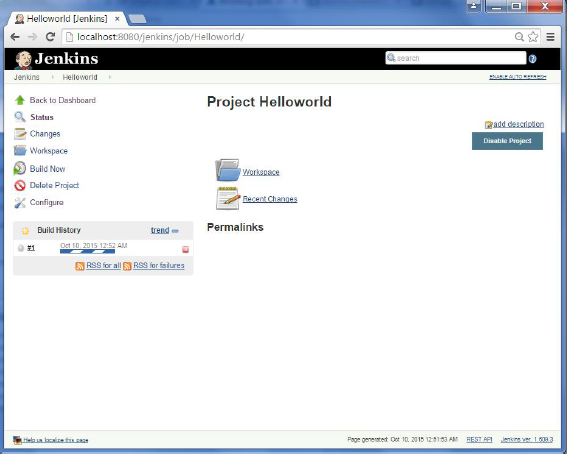
Java HelloWorld



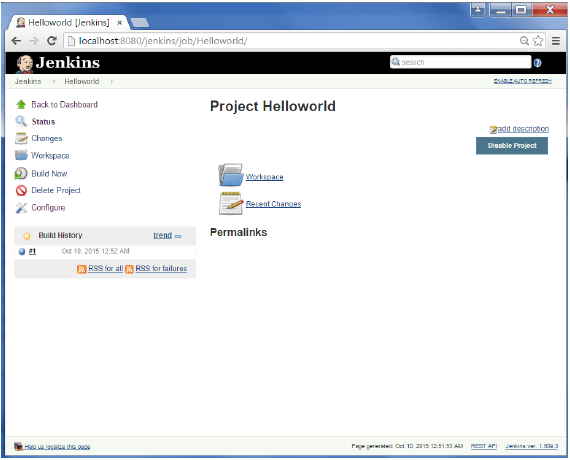
**Step 7** − Once saved, you can click on the Build Now option to see if you have successfully defined the job.



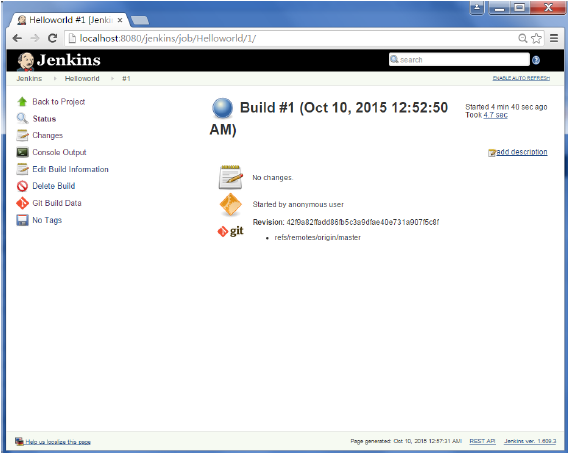
**Step 8** − Once the build is scheduled, it will run. The following Build history section shows that a build is in progress.



**Step 9** − Once the build is completed, a status of the build will show if the build was successful or not. In our case, the following build has been executed successfully. Click on the #1 in the Build history to bring up the details of the build.



**Step 10** − Click on the Console Output link to see the details of the build



Apart from the steps shown above there are just so many ways to create a build job, the options available are many, which what makes Jenkins such a fantastic continuous deployment tool.

## GITHUB

## What is GitHub?

* GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.
* This tutorial teaches you GitHub essentials like repositories, branches, commits, and Pull Requests. You’ll create your own Hello World repository and learn GitHub’s Pull Request workflow, a popular way to create and review code.

#### No coding necessary

* To complete this tutorial, you need a [GitHub.com account](http://github.com/) and Internet access. You don’t need to know how to code, use the command line, or install Git (the version control software GitHub is built on).
* **Tip:** Open this guide in a separate browser window (or tab) so you can see it while you complete the steps in the tutorial.
* At the heart of GitHub is an open source version control system (VCS) called [*Git*](https://git-scm.com/). Git is responsible for everything GitHub-related that happens locally on your computer.
* To use Git on the command line, you'll need to download, install, and configure Git on your computer.
* If you want to work with Git locally, but don't want to use the command line, you can instead download and install the [GitHub Desktop](https://desktop.github.com/) client. For more information, see "[Getting Started with GitHub Desktop](https://help.github.com/desktop/guides/getting-started-with-github-desktop/)."
* If you don't need to work with files locally, GitHub lets you complete many Git-related actions directly in the browser, including:
* [Creating a repository](https://help.github.com/articles/create-a-repo)
* [Forking a repository](https://help.github.com/articles/fork-a-repo)
* [Managing files](https://help.github.com/articles/managing-files-on-github/)
* [Being social](https://help.github.com/articles/be-social)

## Step 1. Create a Repository

A **repository** is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs. We recommend including a README, or a file with information about your project. GitHub makes it easy to add one at the same time you create your new repository. It also offers other common options such as a license file.

Your hello-world repository can be a place where you store ideas, resources, or even share and discuss things with others.

### To create a new repository

1. In the upper right corner, next to your avatar or identity, click and then select **New repository**.
2. Name your repository hello-world.
3. Write a short description.
4. Select **Initialize this repository with a README**.



Click **Create repository**. :tada:

## Step 2. Create a Branch

**Branching** is the way to work on different versions of a repository at one time.

By default your repository has one branch named master which is considered to be the definitive branch. We use branches to experiment and make edits before committing them to master.

When you create a branch off the master branch, you’re making a copy, or snapshot, of master as it was at that point in time. If someone else made changes to the master branch while you were working on your branch, you could pull in those updates.

This diagram shows:

* The master  branch
* A new branch called feature (because we’re doing ‘feature work’ on this branch)
* The journey that feature takes before it’s merged into master

Have you ever saved different versions of a file? Something like:

* story.txt
* story-joe-edit.txt
* story-joe-edit-reviewed.txt

Branches accomplish similar goals in GitHub repositories.

Here at GitHub, our developers, writers, and designers use branches for keeping bug fixes and feature work separate from our master (production) branch. When a change is ready, they merge their branch into master.

### To create a new branch

1. Go to your new repository hello-world.
2. Click the drop down at the top of the file list that says **branch: master**.
3. Type a branch name, readme-edits, into the new branch text box.
4. Select the blue **Create branch** box or hit “Enter” on your keyboard.



Now you have two branches, master and readme-edits. They look exactly the same, but not for long next we’ll add our changes to the new branch.

## Step 3. Make and commit changes

Bravo! Now, you’re on the code view for your readme-edits branch, which is a copy of master. Let’s make some edits.

On GitHub, saved changes are called commits. Each commit has an associated commit message, which is a description explaining why a particular change was made. Commit messages capture the history of your changes, so other contributors can understand what you’ve done and why.

#### Make and commit changes

1. Click the README.md file.
2. Click the pencil icon in the upper right corner of the file view to edit.
3. In the editor, write a bit about yourself.
4. Write a commit message that describes your changes.
5. Click **Commit changes** button.

## commit

## Step 4. Open a Pull Request

Nice edits! Now that you have changes in a branch off of master, you can open a pull request.

Pull Requests are the heart of collaboration on GitHub. When you open a pull request, you’re proposing your changes and requesting that someone review and pull in your contribution and merge them into their branch. Pull requests show diffs, or differences, of the content from both branches. The changes, additions, and subtractions are shown in green and red.

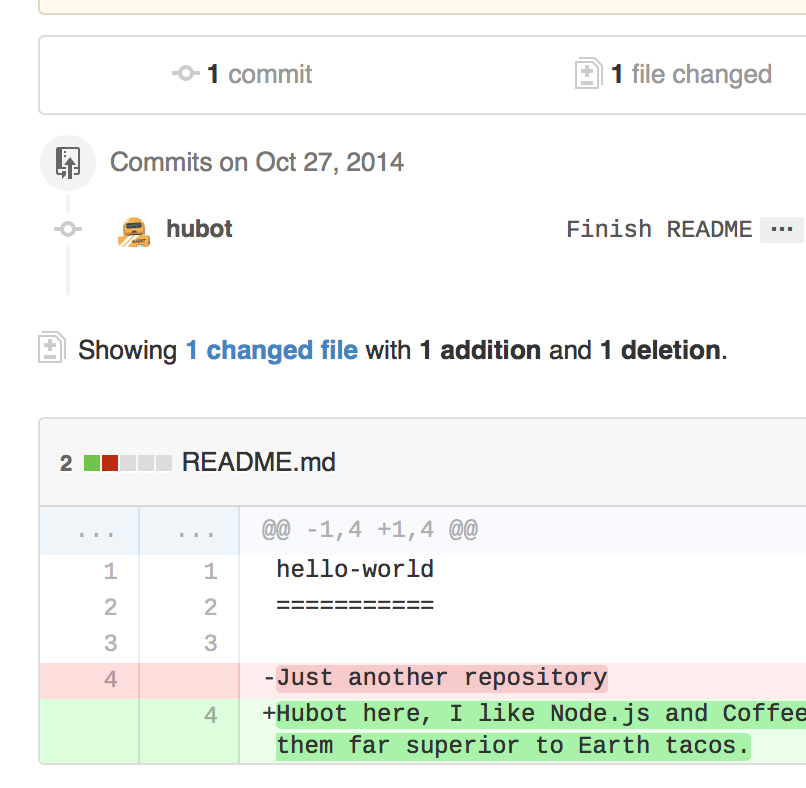
As soon as you make a commit, you can open a pull request and start a discussion, even before the code is finished.

By using GitHub’s [@mention system](https://help.github.com/articles/about-writing-and-formatting-on-github/#text-formatting-toolbar) in your pull request message, you can ask for feedback from specific people or teams, whether they’re down the hall or 10 time zones away.

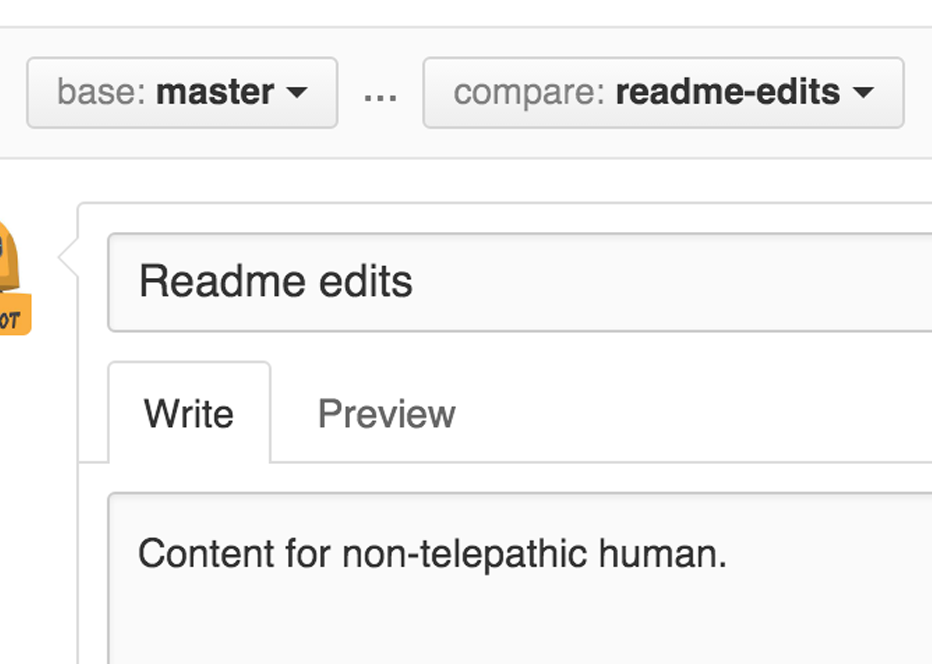
You can even open pull requests in your own repository and merge them yourself. It’s a great way to learn the GitHub flow before working on larger projects.

#### Open a Pull Request for changes to the README

| **Step** | |
| --- | --- |
| Click the  **Pull Request** tab, then from the Pull Request page, click the green **New pull request** button. | |  | |
| In the **Example Comparisons** box, select the branch you made, readme-edits, to compare with master (the original). | |  |
|  |
|  |



When you’re satisfied that these are the changes you want to submit, click the big green **Create Pull Request** button.

When you’re done with your message, click **Create pull request**!

**Tip**: You can use [emoji](https://help.github.com/articles/basic-writing-and-formatting-syntax/#using-emoji) and [drag and drop images and gifs](https://help.github.com/articles/file-attachments-on-issues-and-pull-requests/) onto comments and Pull Requests.

## Step 5. Merge your Pull Request

In this final step, it’s time to bring your changes together – merging your readme-edits branch into the master branch.

1. Click the green **Merge pull request** button to merge the changes into master.
2. Click **Confirm merge**.
3. Go ahead and delete the branch, since its changes have been incorporated, with the **Delete branch** button in the purple box.

### mergedelete

# Branching and Committing

**Branches** are a way to isolate different paths of development, which can then be combined (aka “merged”) into a single branch (often named “master”). Refer to [this short guide](http://gitready.com/beginner/2009/01/25/branching-and-merging.html) or the resources in the [Learning Git](http://ardupilot.org/dev/docs/where-to-get-the-code.html#where-to-get-the-code-learning-git) section for more information.

**Commits** record changes to the code along with a description and the author’s name.

This page describes how to create a branch and add some commits. The instructions show the command line interface but all of the [recommended GUI tools](http://ardupilot.org/dev/docs/git-install.html#git-install) can also perform these functions.

## Creating a Branch

1. open a terminal window and cd to the ardupilot directory of your [clone](http://ardupilot.org/dev/docs/git-clone.html#git-clone) and checkout the master branch
2. git checkout master
3. the new branch will be a copy of the current branch so ensure your [clone’s](http://ardupilot.org/dev/docs/git-clone.html#git-clone) master branch is up-to-date with (aka [rebased on](http://ardupilot.org/dev/docs/git-rebase.html#git-rebase)) [ArduPilot/master](https://github.com/ArduPilot/ardupilot). If you have just [forked](http://ardupilot.org/dev/docs/git-fork.html#git-fork) and [cloned](http://ardupilot.org/dev/docs/git-clone.html#git-clone) your repo then it should already be up-to-date.
4. decide on a branch name and create the new branch. The name is up to you but it can be helpful to choose a short descriptive name. The branch name used for this tutorial is “apm\_git\_tutorial”.
5. git checkout -b apm\_git\_tutorial
6. Change some code. For this tutorial, open the **Tools/GIT\_Test/GIT\_Success.txt** in your preferred text editor, and put your name at the end of the file then save the file
7. See that you have changed the files by checking the status
8. git status
9. **Stage** and **Commit** your work to the branch to record your changes to your [clone](http://ardupilot.org/dev/docs/git-clone.html#git-clone)
10. git add Tools/GIT\_Test/Git\_Success.txt
11. git commit -m 'Tools: added name to GIT\_Success.txt'

In this case, the subject line of the commit is simply “Tools: added name to GIT\_Success.txt” but see [here](http://ardupilot.org/dev/docs/submitting-patches-back-to-master.html#submitting-patches-back-to-master) for more detailed information on conventions for commits that you expect to be integrated into [ArduPilot/master](https://github.com/ArduPilot/ardupilot).

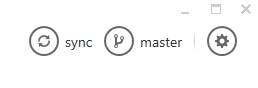
1. **Push** the new branch from your local [clone](http://ardupilot.org/dev/docs/git-clone.html#git-clone) to your [fork](http://ardupilot.org/dev/docs/git-fork.html#git-fork) on GitHub. This will copy your work on your local branch to a new branch on GitHub. Pushing branches is a precondition for collaborating with others on GitHub or for submitting patches back to the official releases. It is assumed “origin” is the remote name of your [fork](http://ardupilot.org/dev/docs/git-fork.html#git-fork) on GitHub
2. git push origin HEAD:apm\_git\_tutorial

Congratulations! This is bulk of the normal process you’d follow when working on code to submit back to the official project. The next step is to [submit a pull request](http://ardupilot.org/dev/docs/submitting-patches-back-to-master.html#submitting-patches-back-to-master) so your changes can be considered for addition to the main project.

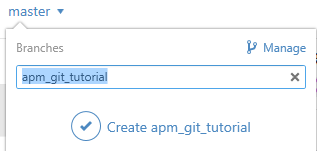
## GitHub GUI specific instructions

In the Github GUI you used to clone the repository, you can create a branch and commit it.

1. Create a branch. In the GitHub for Windows application, click on the ‘master’ button in the upper right corner of the window.

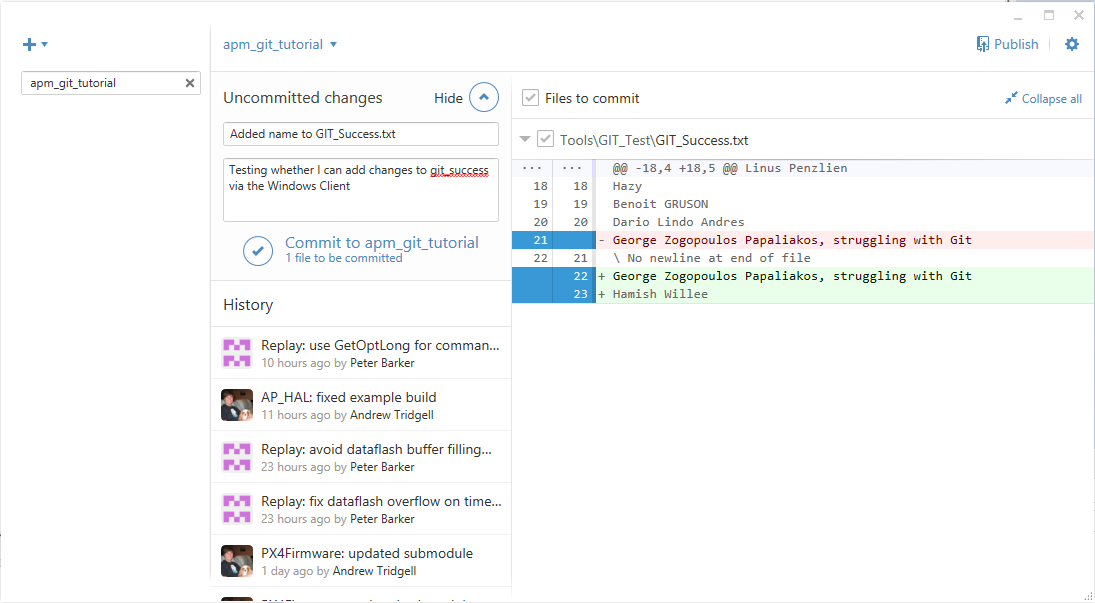
[](http://ardupilot.org/dev/_images/APM-Git-Github-Windows-Branch.jpg)

Enter ‘apm\_git\_tutorial’ and click the “+ create branch: apm\_git\_tutorial” dropdown.

[](http://ardupilot.org/dev/_images/GitHubForWindowsClient_CreateBranch.png)

*Create a new branch in Github for Windows*

1. Change some code. For this tutorial, open the **Tools/GIT\_Test/GIT\_Success.txt** in your preferred text editor, and put your name at the end of the file. Save the file.
2. The Git for Windows client shows the changed file and has a place where you can enter a summary and description of the change. For the purpose of this tutorial, you can just use a single line stating: “Tools: added name to GIT\_Success.txt”

[](http://ardupilot.org/dev/_images/GitHubForWindowsClient_CommitingChange.png)

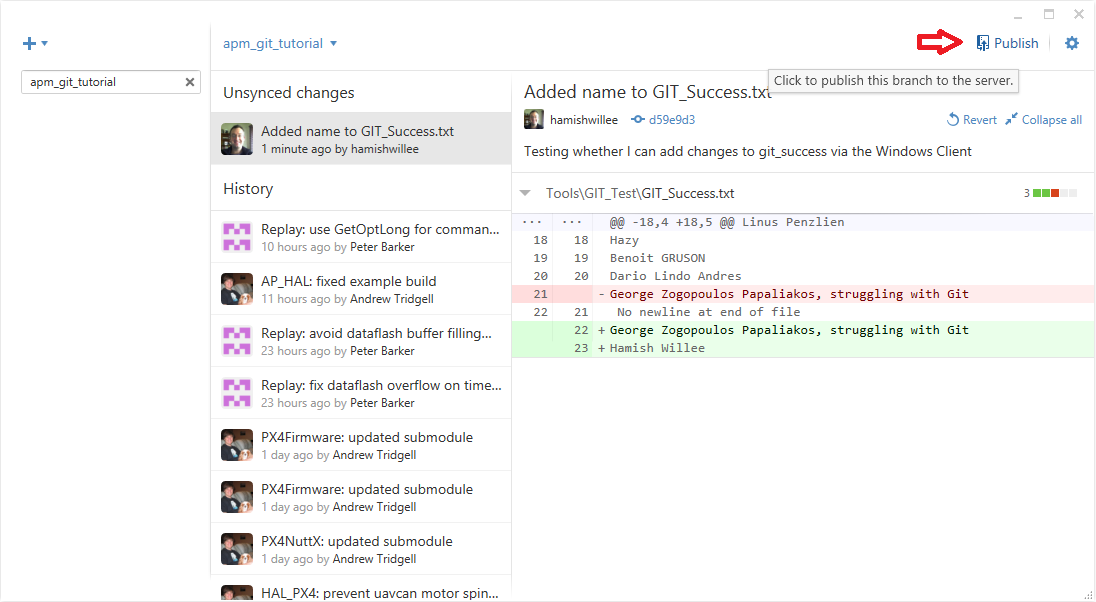
*Github for windows: Commit Change*

**Note**

**Please see**[**Submitting Patches Back to Master**](http://ardupilot.org/dev/docs/submitting-patches-back-to-master.html#submitting-patches-back-to-master)**for further**

information regarding conventions for committing work that you expect to be integrated into the official releases (this will include a much more detailed commit message).

1. Commit your work to the branch by pressing the **Commit to apm\_git\_tutorial** link.
2. Push your local branch to GitHub (pushing branches is a precondition for collaborating with others on GitHub or for submitting patches back to the official releases). In the client you can do this by pressing the **Publish** link:

[](http://ardupilot.org/dev/_images/GitHubForWindowsClient_Publish_Push.png)

*GitHub for Windows Client: Pushing changes*

AMAZON EC2

What Is Amazon EC2?

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

Features of Amazon EC2:

* Virtual computing environments, known as instances.
* Preconfigured templates for your instances, known as Amazon Machine Images (AMIs), that package the bits you need for your server (including the operating system and additional software).
* Various configurations of CPU, memory, storage, and networking capacity for your instances, known as instance types.
* Secure login information for your instances using key pairs (AWS stores the public key, and you store the private key in a secure place).
* Storage volumes for temporary data that's deleted when you stop or terminate your instance, known as instance store volumes.
* Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as Amazon EBS volumes.
* Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as regions and Availability Zones.
* A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using security groups.
* Static IPv4 addresses for dynamic cloud computing, known as Elastic IP addresses.
* Metadata, known as tags, that you can create and assign to your Amazon EC2 resources.
* Virtual networks you can create that are logically isolated from the rest of the AWS cloud, and that you can optionally connect to your own network, known as virtual private clouds (VPCs)

## How to Get Started with Amazon EC2

First, you need to get set up to use Amazon EC2. After you are set up, you are ready to complete the Getting Started tutorial for Amazon EC2. Whenever you need more information about an Amazon EC2 feature, you can read the technical documentation.

**Get Up and Running**

* [Setting Up with Amazon EC2](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/get-set-up-for-amazon-ec2.html)
* [Getting Started with Amazon EC2 Linux Instances](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html)

**Basics**

* [Instances and AMIs](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instances-and-amis.html)
* [Regions and Availability Zones](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html)
* [Instance Types](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html)
* [Tags](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Using_Tags.html)

**Networking and Security**

* [Amazon EC2 Key Pairs](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html)
* [Security Groups](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html)
* [Elastic IP Addresses](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html)
* [Amazon EC2 and Amazon VPC](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-vpc.html)

**Storage**

* [Amazon EBS](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AmazonEBS.html)
* [Instance Store](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/InstanceStorage.html)

**Working with Linux Instances**

* [Remote Management (Run Command)](https://docs.aws.amazon.com/systems-manager/latest/userguide/execute-remote-commands.html)
* [Tutorial: Install a LAMP Web Server on Amazon Linux 2](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-lamp-amazon-linux-2.html)
* [Tutorial: Configure Apache Web Server on Amazon Linux 2 to Use SSL/TLS](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/SSL-on-an-instance.html)
* [Getting Started with AWS: Hosting a Web App for Linux](https://docs.aws.amazon.com/gettingstarted/latest/wah-linux/)

### AUTOMATION OF A WEBSITE THROUGH JENKINS

### STEPS :

### Launch a EC2 instance and connect it through putty or virtual machine.

### <https://cdn1.howtodoinjava.com/wp-content/uploads/2017/07/EC2_14.jpg>

### This is the screenshot of launched EC2 instance.

### <https://cdn2.howtodoinjava.com/wp-content/uploads/2017/07/putty7.jpg>

### This is the screenshot of connected EC2 instance through Putty.

### Install Apache in ubuntu.

### Command – sudo apt-get update

### Sudo apt-get install apache

### Install Jenkins in ubuntu.

### Command – sudo apt-get update

### wget -q -O - <https://pkg.jenkins.io/debian/jenkins->

### ci.org.key | sudo apt-key add –

### sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable

### binary/etc/apt/sources.list.d/jenkins.list'

### sudo apt-get update

### sudo apt-get install Jenkins

### Connect Jenkins by entering public ip : 8080 in internet explorer.

### Accessing Jenkins

### Make a directory in local of any name.

### Make a repository of any name in github.

### new-repo-form

### Make a job in Jenkins of any name.

### Helloworld

### Clone your repository from github by using

### Command – git clone <url of your repository from github>

### 

### Change your directory to repository directory and download a static website here by using

### Command – sudo wget <link address of static template>

### Unzip your template file by using

### Command – sudo unzip <file name>

### If not working then Install unzip in your local by using

### Command – sudo apt-get update

### Sudo apt-get install unzip

### Use command – git add .

### Then use command – git status (you will found the color of all files are

### green and ready to commit)

### Use command – git commit -m “message which you want”

### Use command – git push (it will ask you your username and password of github one by one)

### Install “Github” plugin from manage plugin in Jenkins.

### Git Plugin

### Go to configure part of Jenkins and source code management part.

### Now copy the url of your repository from github and paste the url in source code management section.

### Git Repository

### Now give some commands in build\_Add build step\_execute Shell

### Command - #! /bin/bash

### cp -R /var/lib/jenkins/workspace/viva /var/www/html

### exit

### Save

### Use build now option and your build will be successful in console output.

### Open a new tab in internet explorer by typing

### public\_ip/repository\_name