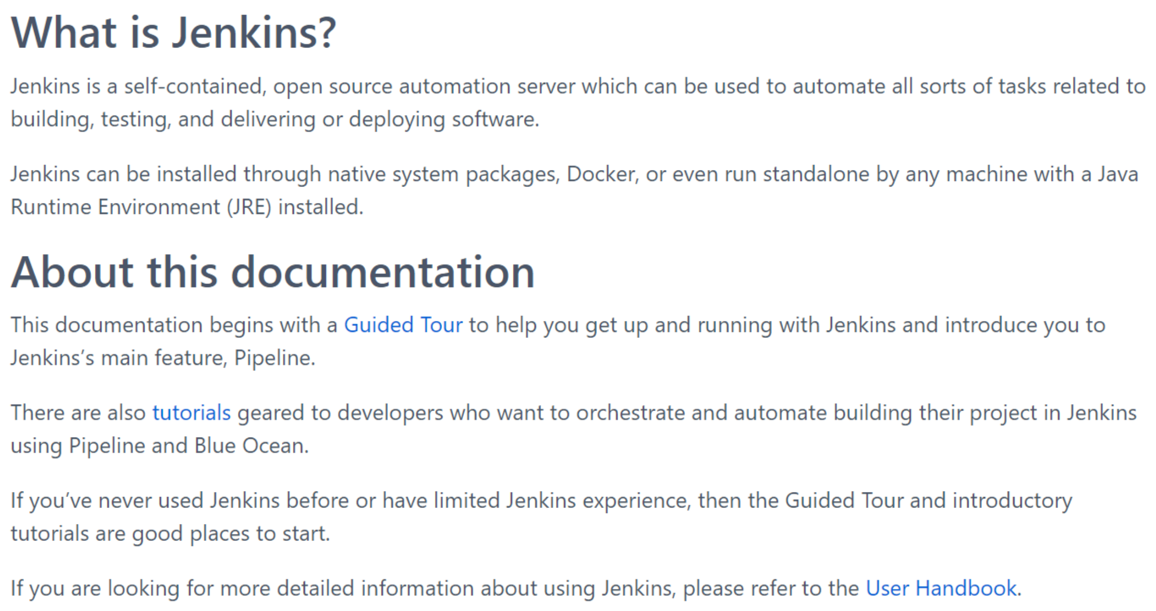
# Jenkins

# https://www.jenkins.io/



## 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.

## About this documentation

This documentation begins with a Guided Tour to help you get up and running with Jenkins and introduce you to Jenkins's main feature, Pipeline.

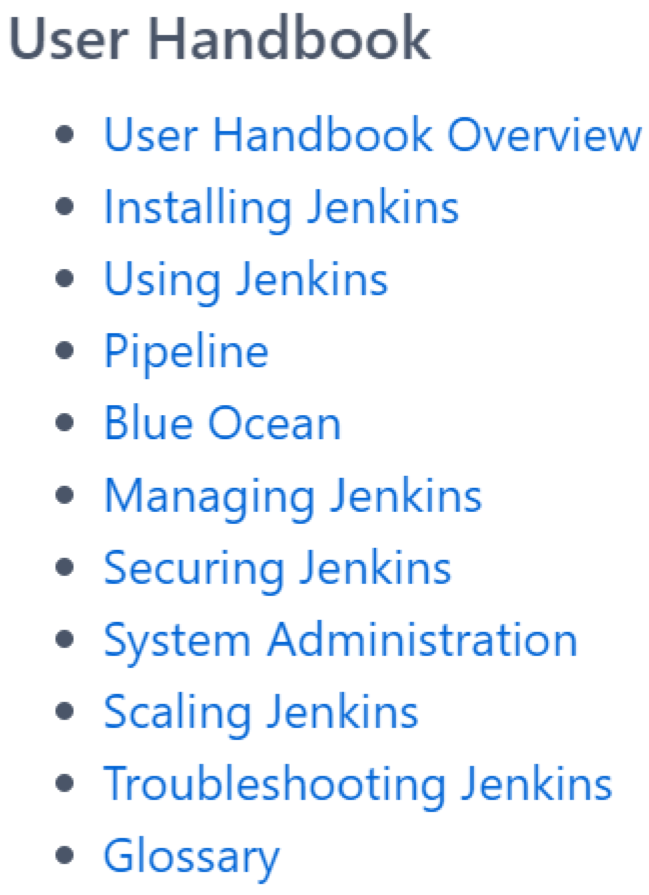
There are also tutorials geared to developers who want to orchestrate and automate building their project in Jenkins using Pipeline and Blue Ocean.

If you've never used Jenkins before or have limited Jenkins experience, then the Guided Tour and introductory tutorials are good places to start.

If you are looking for more detailed information about using Jenkins, please refer to the User Handbook.

**Explore and document different documentation needs (e.g user manuals, installation/deployment documents, development/extension documents, tutorials, etc)**

1. **User Manuals**



User Handbook

- User Handbook Overview
- Installing Jenkins
- Using Jenkins
- Pipeline
- Blue Ocean
- Managing Jenkins
- Securing Jenkins
- System Administration
- Scaling Jenkins
- Troubleshooting Jenkins
- Glossary

## 2. Installation Documents

## Installing Jenkins

- Docker
- Kubernetes
- Linux
- macOS
- Windows
- Other Systems
- WAR file
- Other Servlet Containers
- Offline Installations
- Initial Settings

### Prerequisites

For this tour, you will require:

- A machine with:
  - 256 MB of RAM, although more than 2 GB is recommended
  - 10 GB of drive space (for Jenkins and your Docker image)
- The following software installed:
  - Java 11 or Java 17
  - Docker (navigate to **Get Docker** site to access the Docker download that's suitable for your platform)

### Download and run Jenkins

1. Download Jenkins Generic Java package (.war)
2. Open up a terminal in the download directory
3. Run `java -jar jenkins.war --httpPort=8080`
4. Browse to `http://localhost:8080`
5. Follow the instructions to complete the installation

When the installation is complete, you can start putting Jenkins to work!

# 3. Development/Extension Documents

## Extend Jenkins

Jenkins is an extensible automation server with more than 1800 plugins providing integrations for hundreds of tools and services. The resources below will teach you how to *extend* Jenkins with new features.

If you want to *use* Jenkins's existing functionality and plugin features only, please refer to the Jenkins User Documentation instead.

### Create a Plugin - Tutorial

Get started with Jenkins development by writing a simple plugin.

### Improve a Plugin - Tutorial

Improve a Jenkins plugin to make it easier to maintain.

### How-To Guides

These guides explain how to solve a specific problem.

### Reference Documentation

The Jenkins developer reference documentation is organized by topics.

### Extensions Index

An index of all extension points available in core and plugins and their implementations.

# 4. Tutorials

## Tutorials overview

This part of the Jenkins User Documentation contains a series of introductory tutorials to help you begin building your applications in an automated fashion with Jenkins.

If you're a developer who wants to improve your understanding of Continuous Integration (CI) / Continuous Delivery (CD) concepts, or you might already be familiar with these concepts but don't yet know how to implement them in Jenkins, then these tutorials are a great place to start.

### Getting started with Jenkins

- Guided Tour to Jenkins

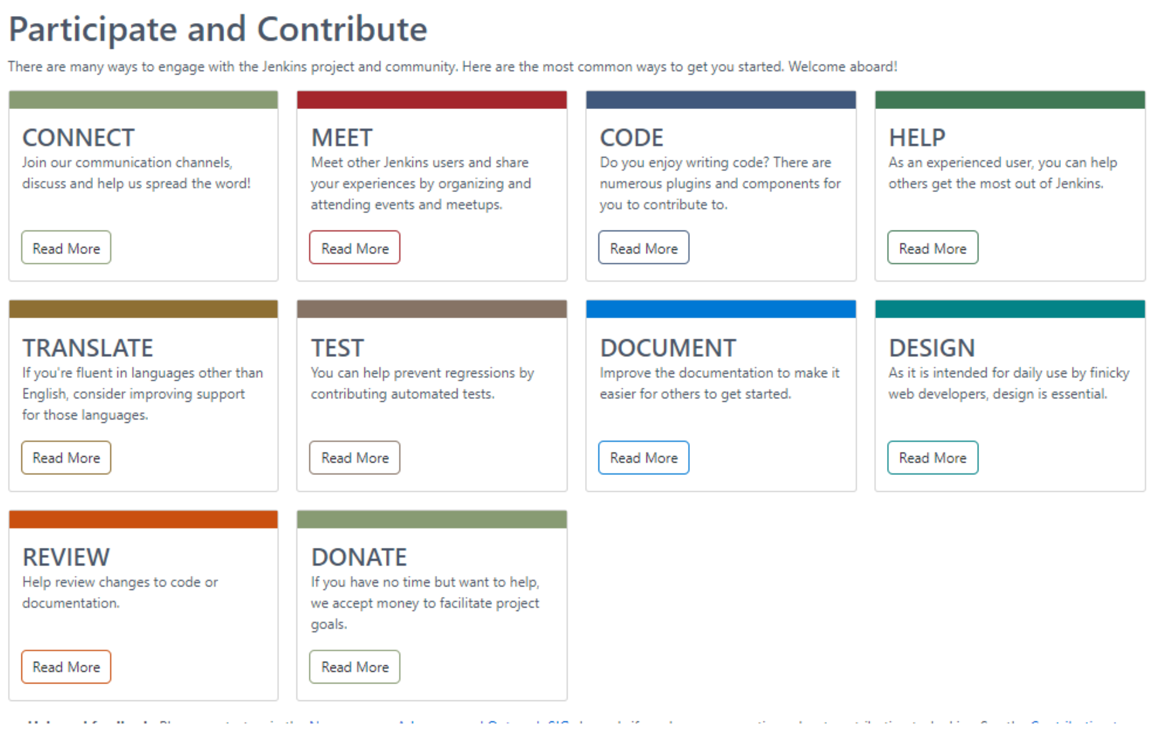
### Pipeline

The following tutorials show how to use key features of Jenkins to facilitate implementing CI/CD processes to build your applications:

- Getting started with Jenkins Pipeline
- Publishing HTML Reports in Pipeline (Declarative Pipeline, Scripted Pipeline)
- Sending Notifications in Pipeline (Declarative Pipeline, Scripted Pipeline)
- End-to-End Multibranch Pipeline Project Creation
- Creating a shared library
- Faster Pipelines with the Parallel Test Executor Plugin
- Converting Conditional Build Steps to Jenkins Pipeline
- Pipeline Development Tools

**Explore and document different process requirements (e.g Versioning and release Management, Issue Life cycle management, Open-source contribution requirements for developers, review process, etc)**

---



## Participate and Contribute

There are many ways to engage with the Jenkins project and community. Here are the most common ways to get you started. Welcome aboard!

**CONNECT**
Join our communication channels, discuss and help us spread the word!
Read More

**MEET**
Meet other Jenkins users and share your experiences by organizing and attending events and meetups.
Read More

**CODE**
Do you enjoy writing code? There are numerous plugins and components for you to contribute to.
Read More

**HELP**
As an experienced user, you can help others get the most out of Jenkins.
Read More

**TRANSLATE**
If you're fluent in languages other than English, consider improving support for those languages.
Read More

**TEST**
You can help prevent regressions by contributing automated tests.
Read More

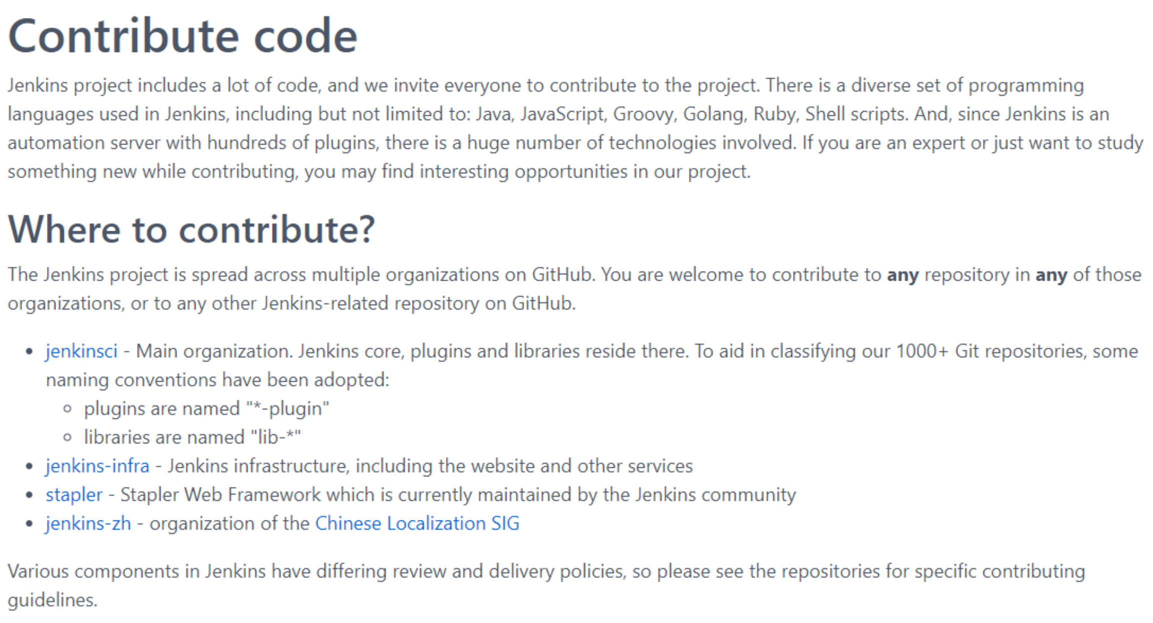
**DOCUMENT**
Improve the documentation to make it easier for others to get started.
Read More

**DESIGN**
As it is intended for daily use by finicky web developers, design is essential.
Read More

**REVIEW**
Help review changes to code or documentation.
Read More

**DONATE**
If you have no time but want to help, we accept money to facilitate project goals.
Read More

## 1. Open Source Contributions



## Contribute code

Jenkins project includes a lot of code, and we invite everyone to contribute to the project. There is a diverse set of programming languages used in Jenkins, including but not limited to: Java, JavaScript, Groovy, Golang, Ruby, Shell scripts. And, since Jenkins is an automation server with hundreds of plugins, there is a huge number of technologies involved. If you are an expert or just want to study something new while contributing, you may find interesting opportunities in our project.

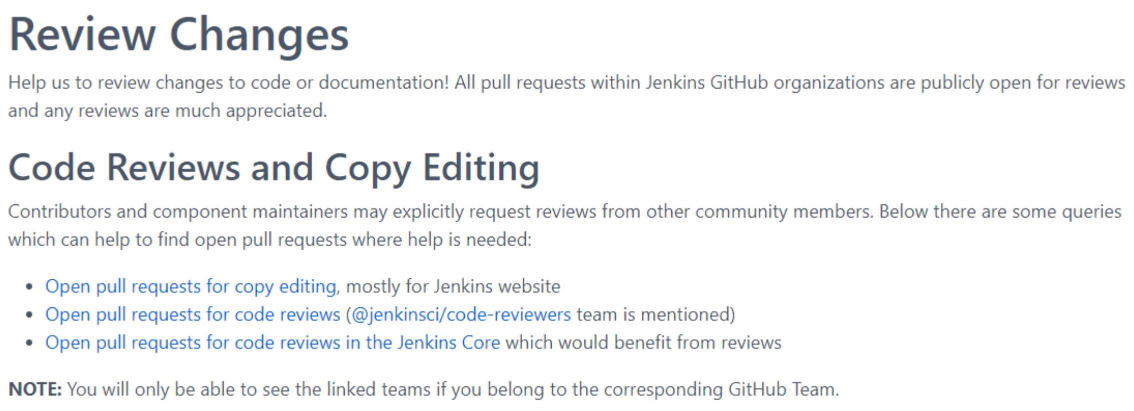
### Where to contribute?

The Jenkins project is spread across multiple organizations on GitHub. You are welcome to contribute to **any** repository in **any** of those organizations, or to any other Jenkins-related repository on GitHub.

- jenkinsci - Main organization. Jenkins core, plugins and libraries reside there. To aid in classifying our 1000+ Git repositories, some naming conventions have been adopted:
  - plugins are named "*-plugin"
  - libraries are named "lib-*"
- jenkins-infra - Jenkins infrastructure, including the website and other services
- stapler - Stapler Web Framework which is currently maintained by the Jenkins community
- jenkins-zh - organization of the Chinese Localization SIG

Various components in Jenkins have differing review and delivery policies, so please see the repositories for specific contributing guidelines.

# 2. Review Code



## Review Changes

Help us to review changes to code or documentation! All pull requests within Jenkins GitHub organizations are publicly open for reviews and any reviews are much appreciated.

### Code Reviews and Copy Editing

Contributors and component maintainers may explicitly request reviews from other community members. Below there are some queries which can help to find open pull requests where help is needed:

- Open pull requests for copy editing, mostly for Jenkins website
- Open pull requests for code reviews (@jenkinsci/code-reviewers team is mentioned)
- Open pull requests for code reviews in the Jenkins Core which would benefit from reviews

**NOTE:** You will only be able to see the linked teams if you belong to the corresponding GitHub Team.

## Explore and document different architectural and component level modularity requirements (e.g. Micro-services, Modular folder structure, class/component reusability, extensibility, Linters for better code quality, etc

### 1. Architecture

### Architecture

⚠ This section is a work in progress. Want to help? Check out the jenkinsci-docs mailing list. For other ways to contribute to the Jenkins project, see this page about participating and contributing.

This section provides a high level overview of the Jenkins architecture.

The audience is Java developers familiar with Jenkins (as users) who want to understand how Jenkins works internally.

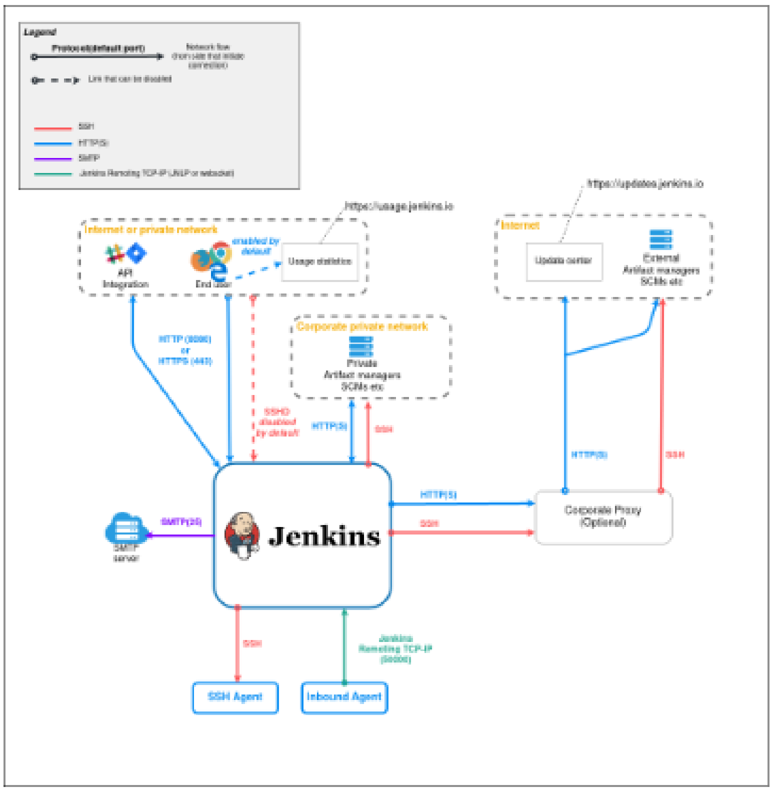
#### Dataflow diagram

This dataflow represents Jenkins as a "box". The goal is to have a view of network flow that goes in and out.

This diagram is a high level view centered on Jenkins core.

Please note that:

- some plugins such as the LDAP plugin can contribute to add new network flow.
- All port numbers can be changed by configuration. The port in the diagram is the default port
- Jenkins can be configured to use TLS (HTTPS)
- It's not mandatory to have all agent types, there is one instance of each agent to illustrate communication types


Legend
Protocol(default port)
Network flow (from side that initiate connection)
Link that can be disabled
SSH
HTTP(S)
SMTP
Jenkins Remoting TCP-IP (JNLP or websocket)
https://usage.jenkins.io
https://updates.jenkins.io
Internet or private network
API Integration
End user
enabled by default
Usage statistics
Internet
Update center
External Artifact managers SCMs etc
HTTP (8080) or HTTPS (443)
Corporate private network
Private Artifact managers SCMs etc
SSHD disabled by default
HTTP(S)
SSH
HTTP(S)
SSH
HTTP(S)
SSH
Corporate Proxy (Optional)
SMTP(25)
SMTP server
Jenkins
SSH
Jenkins Remoting TCP-IP (50000)
SSH Agent
Inbound Agent

The source of this diagram is on github, to be opened with https://app.diagrams.net.

### 2. Extensibility

# Extensibility

This section is a work in progress. Want to help? Check out the jenkinsci-docs mailing list. For other ways to contribute to the Jenkins project, see this page about participating and contributing.

Jenkins is designed with extensibility in mind, both internally (core) and with plugins. Extensibility in Jenkins is accomplished by combining the concepts discussed below.

## Extension annotation

Extension is an annotation that allows Jenkins to discover classes, instantiate them, and register them in global lists of implementations of their supertypes and interfaces. This works for implementations both directly in core, and contributed by plugins, so that plugins can provide implementations of extension points defined in core (or other plugins). Whenever Jenkins needs to provide a list, e.g. of security realm implementations (LDAP, Jenkins user database, etc.) it can query for all subtypes of `SecurityRealm` that are annotated with `@Extension`.

## These Non functional requirements make the code better by:

1. **Usability**

**Learnability -** Degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use. The documents tutorials user and insallationguides make it easier

**Operability -** Degree to which a product or system has attributes that make it easy to operate and control. The user guides make it operable

**Accessibility -** Degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use. As the pre requisites mentioned are simple and easy, it is easily accessible

**2.Portability**

**Installability -** Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment. The installation guides make it easier to do so.

.