

Jenkins and Blue Ocean

Lab 6

Create a Pipeline that executes Exercise 2 test automatically.

The pipeline must have at least the following stages:

1. Build
2. Flash LPCXpresso
3. RobotFramework tests

Note that in addition to returning required files to Oma you also need to show that your pipeline works to the instructor.

Setup

In this exercise we run Jenkins and Gogs (a Git server) in containers. On the first run we need to configure both Jenkins and Gogs. After the initial setup the container can be stopped and started at will. The settings are retained in named volumes so there is no need to reconfigure the services.

To keep the URLs consistent both in containers and on the host computer you need to edit `/etc/hosts`. On linux the file path is `/etc/hosts` (what a surprise!). On windows the file path is:

`C:\Windows\System32\drivers\etc\hosts`. Note that you need administrator privileges to edit the file. Add the following lines to `/etc/hosts`:

```
# Reroute jenkins and gogs to localhost
127.0.0.1 jenkins
127.0.0.1 gogs
```

Windows version of Docker installs Docker compose automatically. If you use Linux you need to install Docker compose manually after installing Docker.

Copy Dockerfile and docker-compose.yml to the same directory. Open command prompt/terminal, go to the directory and execute:

```
docker-compose up
```

Now your containers are up and running and you can perform the initial configuration of Jenkins and Gogs. Following chapters explain how to setup the services.

When you want to stop your containers just execute:

```
docker-compose down
```

You can start the contained in detached mode by executing:

```
docker-compose up --detached
```

In detached mode you won't see the console messages. You don't really need the console messages after the initial setup so once your containers are setup start the containers in detached mode.

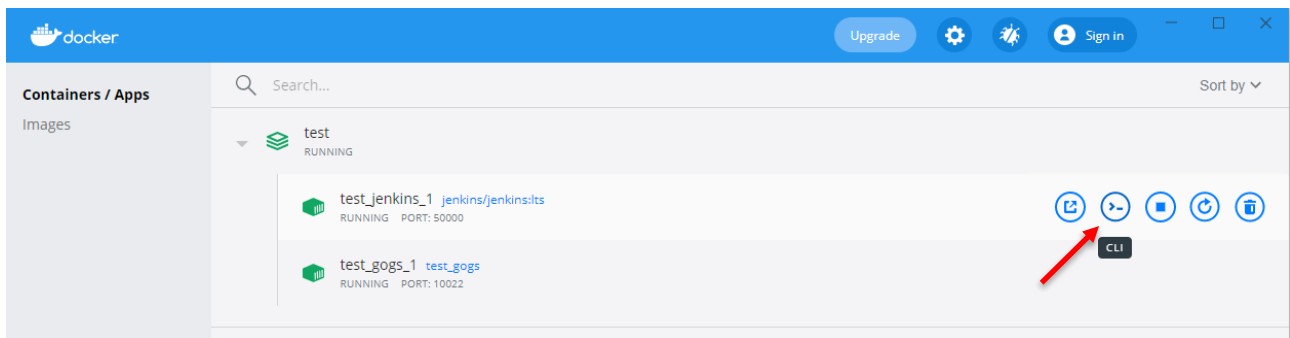
Navigate to `jenkins:8080` in your browser.

```
Windows PowerShell
```

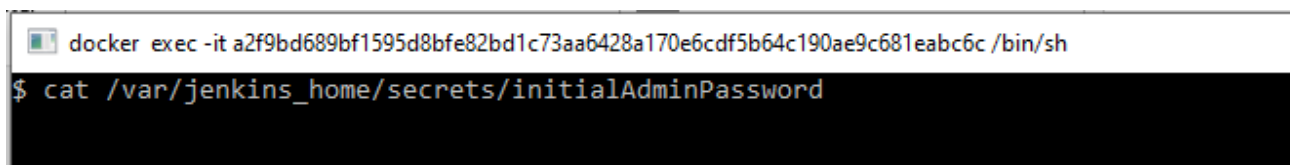
```
jenkins_1 | 2021-04-20 06:26:45.473+0000 [id=31] INFO jenkins.InitReactorRunner$1#onAttained: System config lo  
ad ed  
jenkins_1 | 2021-04-20 06:26:45.474+0000 [id=43] INFO jenkins.InitReactorRunner$1#onAttained: System config ad  
apt ed  
jenkins_1 | 2021-04-20 06:26:45.475+0000 [id=31] INFO jenkins.InitReactorRunner$1#onAttained: Loaded all jobs  
jenkins_1 | 2021-04-20 06:26:45.475+0000 [id=33] INFO jenkins.InitReactorRunner$1#onAttained: Configuration fo  
r all jobs updated  
jenkins_1 | 2021-04-20 06:26:45.485+0000 [id=56] INFO hudson.model.AsyncPeriodicWork#$lambda$doRun$: Started D  
ownload metadata  
jenkins_1 | 2021-04-20 06:26:45.494+0000 [id=56] INFO hudson.util.Retrier$start: Attempt #1 to do the action c  
heck updates server  
jenkins_1 | 2021-04-20 06:26:45.949+0000 [id=38] INFO jenkins.install.SetupWizard#init:  
  
jenkins_1 | *****  
jenkins_1 | *****  
jenkins_1 | *****  
jenkins_1 | *****  
jenkins_1 | Jenkins initial setup is required. An admin user has been created and a password generated.  
jenkins_1 | Please use the following password to proceed to installation:  
jenkins_1 | d358bcd5f4dc44f89112b785211dd56c  
jenkins_1 | This may also be found at: /var/jenkins-home/secrets/initialAdminPassword  
jenkins_1 | *****  
jenkins_1 | *****  
jenkins_1 | *****  
jenkins_1 | *****  
jenkins_1 | 2021-04-20 06:26:54.277+0000 [id=43] INFO jenkins.InitReactorRunner$1#onAttained: Completed initia  
lization  
jenkins_1 | 2021-04-20 06:26:54.296+0000 [id=221] INFO hudson.WebAppMain$3run: Jenkins is fully up and running
```

Note that the browser will complain about insecure connection. Just ignore the warning because we are actually connecting to localhost.

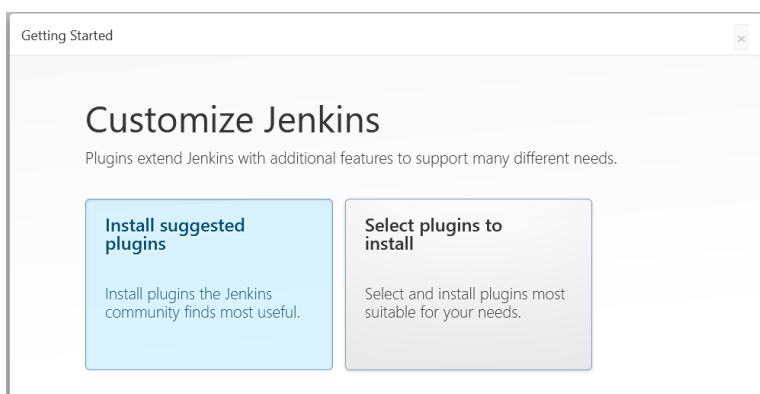
If you can't find the initial admin password from the logs you can run a console in the container. See below.



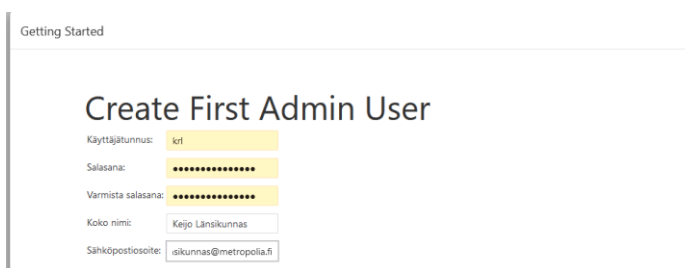
Type command in the console. Note that there is no tab-completion or command history in the Jenkins containers shell.



After entering the initial password, you get the following screen. Select install suggested plugins.



After the plugins have been installed you need to create the first admin user. It is important to keep this password safe because the initial password is a onetime password that will not work after this stage.



After creating Admin user, you need to configure the URL used to access Jenkins. Accept the default URL.

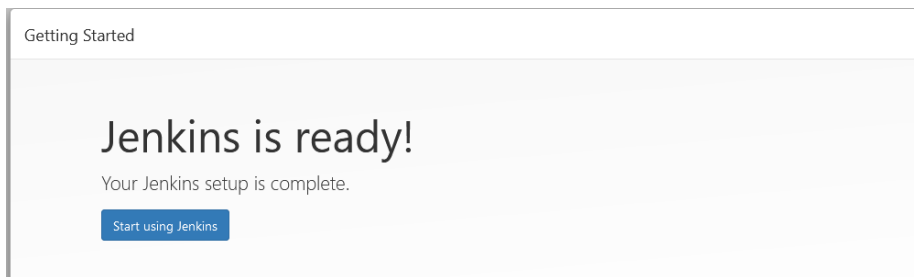
Getting Started

Instance Configuration


Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.





At this point Jenkins is NOT ready. Click **Start using Jenkins** and install some additional plugins according to the instructions below. Depending on your browser settings you may see part of Jenkins UI in English and part in some other language.

 **Jenkins**

Dashboard ▾

 Uusi Item

 Käyttäjät

 Käännöshistoria

 Hallitse Jenkinsia

 My Views

 Lockable Resources

 New View

Käännösjono ^

Ei käännöksiä jonossa.

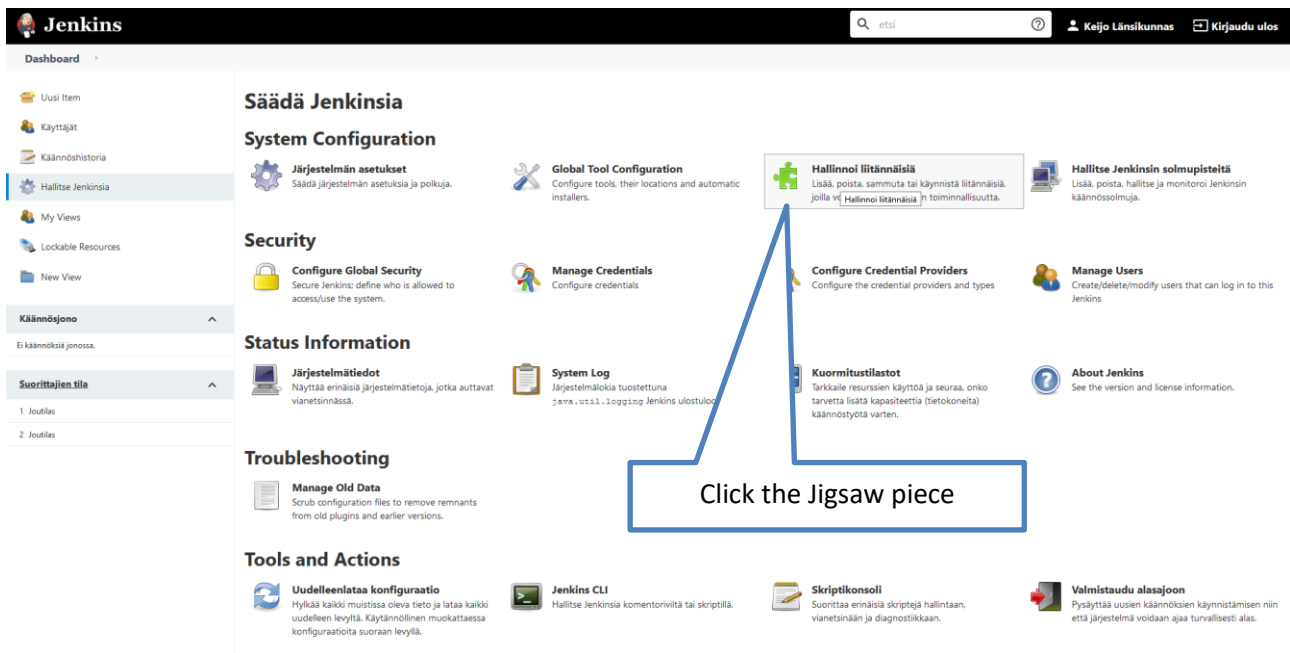
Suorittajien tila ^

1 Joutilas

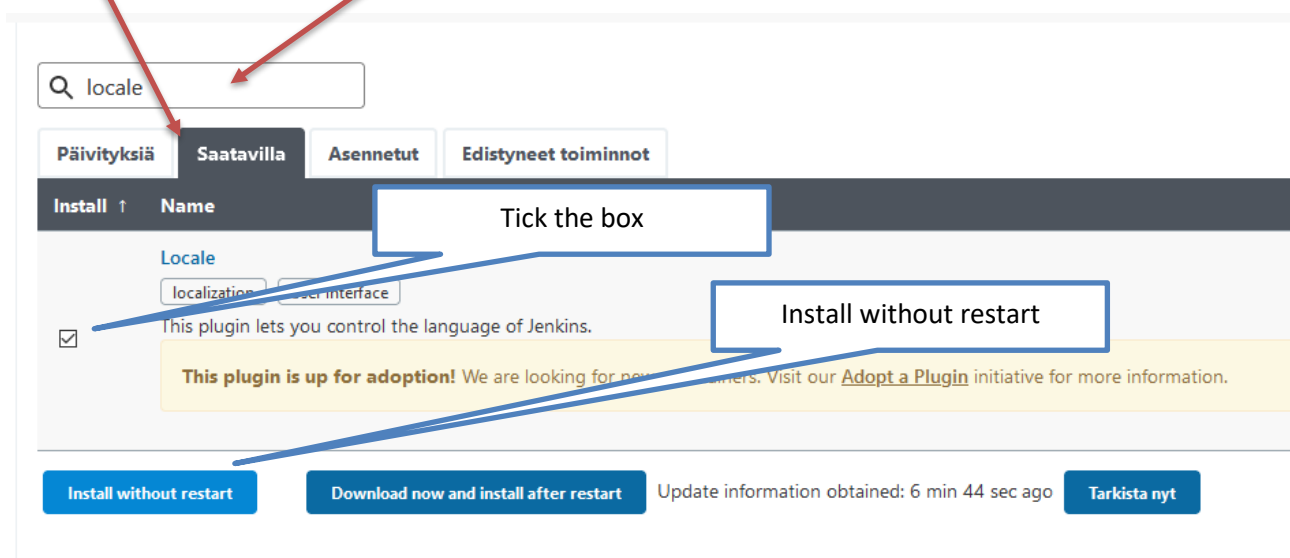
2 Joutilas

Click the gear icon

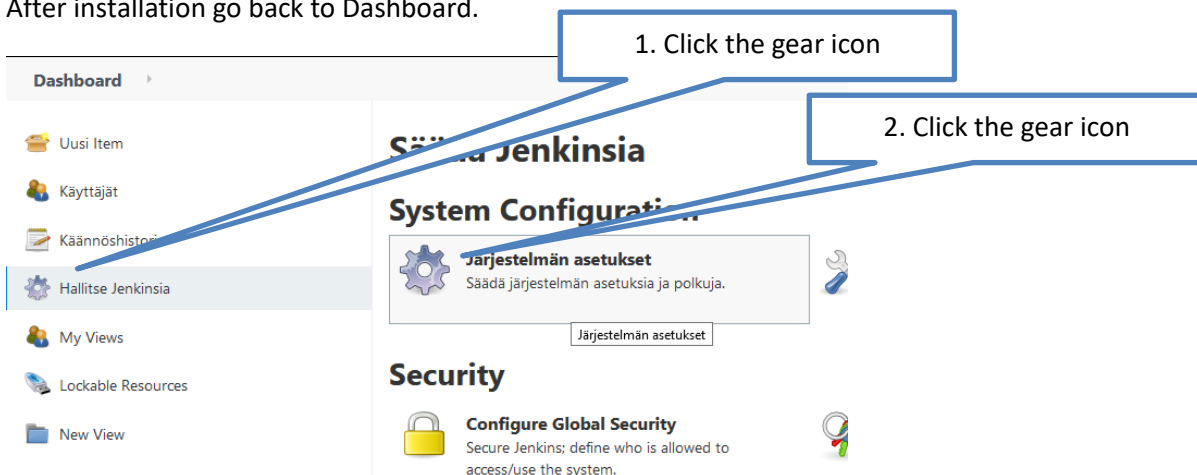
What a mess!



Select second tab and type locale in the search box.



After installation go back to Dashboard.



Scroll down to **Locale** and do the following changes:

Locale

Default Language

en

☒ Ignore browser preference and force this language to all users

Click **Apply** at the bottom of the page.

Now your Jenkins UI is consistently in English.

The screenshot shows the Jenkins Dashboard with the 'Manage Jenkins' section selected in the left sidebar. The main content area is titled 'Manage Jenkins' and contains several sub-sections:

- System Configuration**
 - Configure System**: Configure global settings and paths.
 - Global Tool Configuration**: Configure tools, their locations and automatic installers.
 - Manage Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
 - Manage Nodes and Clouds**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Security**
 - Configure Global Security**: Secure Jenkins: define who is allowed to access/use the system.
 - Manage Credentials**: Configure credentials.
 - Configure Credential Providers**: Configure the credential providers and types.
 - Manage Users**: Create/delete/modify users that can log in to this Jenkins.
- Status Information**
 - System Information**: Displays various environmental information to assist trouble-shooting.
 - System Log**: System log captures output from `java.util.logging` output related to Jenkins.
 - Load Statistics**: Check your resource utilization and see if you need more computers for your builds.
 - About Jenkins**: See the version and license information.
- Troubleshooting**
 - Manage Old Data**: Scrub configuration files to remove remnants from old plugins and earlier versions.
- Tools and Actions**
 - Reload Configuration from Disk**: Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.
 - Jenkins CLI**: Access/manage Jenkins from your shell, or from your script.
 - Script Console**: Executes arbitrary script for administration/trouble-shooting/diagnostics.
 - Prepare for Shutdown**: Stops executing new builds, so that the system can be eventually shut down safely.

Go to Manage Plugins, select **available** and type blue ocean in the search box.

Select: Blue ocean – BlueOcean Aggregator

The screenshot shows the 'Manage Plugins' page in Jenkins. The search box contains 'blue ocean'. The 'Available' tab is selected, and the search results are displayed in a table:

Install	Name
<input checked="" type="checkbox"/>	Blue Ocean External Site/Tool Integrations User Interface BlueOcean Aggregator

Click **Install without restart** at the bottom of the page.

Install without restart

Download now and install after restart

Update information obtained: 10 min ago

Check now

Wait patiently, the aggregator installs quite a few plugins.

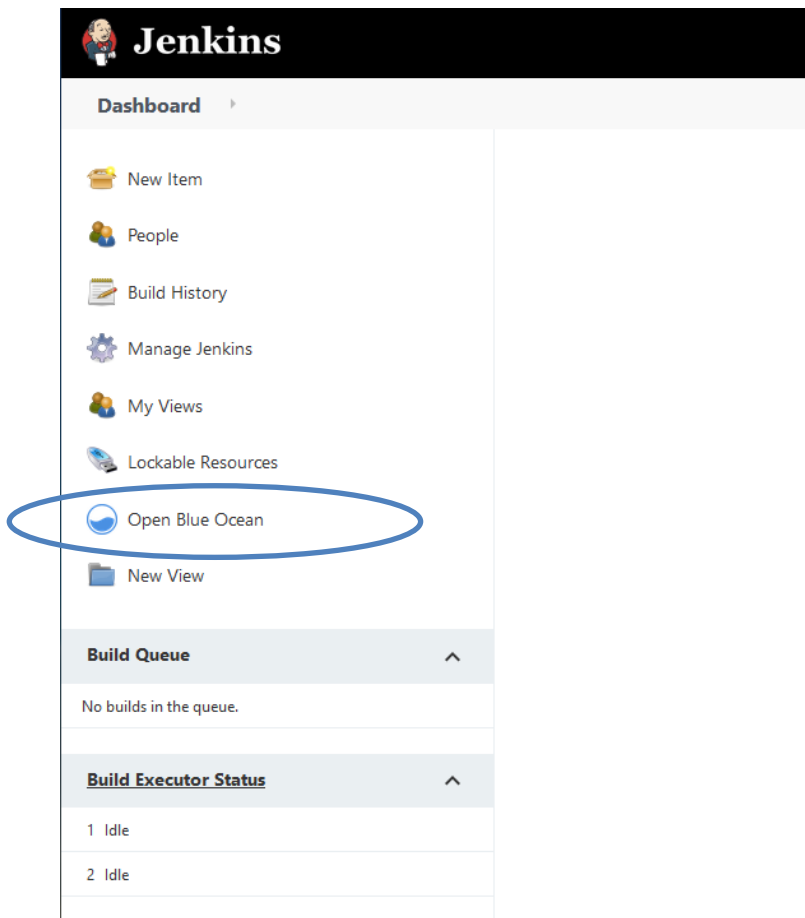
Mailer	Success
Loading plugin extensions	Success
Locale	Success
Loading plugin extensions	Success
Design Language	Success
Blue Ocean Core JS	Success
Common API for Blue Ocean	Success
REST API for Blue Ocean	Success
Pub-Sub "light" Bus	Success
Pipeline SCM API for Blue Ocean	Success
HTML Publisher	Success
Variant	Success
Web for Blue Ocean	Success
JWT for Blue Ocean	Success
Favorite	Success
REST Implementation for Blue Ocean	Success
Pipeline implementation for Blue Ocean	Success
GitHub Pipeline for Blue Ocean	Success
Git Pipeline for Blue Ocean	Success
Config API for Blue Ocean	Success
Authentication Tokens API	Pending
Handy Uri Templates 2.x API	Pending
Bitbucket Branch Source	Pending
Bitbucket Pipeline for Blue Ocean	Pending
Dashboard for Blue Ocean	Pending
Personalization for Blue Ocean	Pending
Display URL for Blue Ocean	Pending
Server Sent Events (SSE) Gateway	Pending
Events API for Blue Ocean	Pending
Blue Ocean Pipeline Editor	Pending
i18n for Blue Ocean	Pending
Autofavorite for Blue Ocean	Pending
Blue Ocean	Pending
Loading plugin extensions	Pending

[Go back to the top page](#)
(you can start using the installed plugins right away)

☐ Restart Jenkins when installation is complete and no jobs are running

When the installation finishes, you can return to dashboard.

If installation was successful, you will see a new icon in the dashboard.



Jenkins needs a repository to connect to so the next step is to configure Gogs. Gogs is a light-weight Git server that is easy to manage.

Gogs setup

Initial setup is done on a single page. Settings should be as shown below. These settings can't be changed from GUI later! Open a new tab in browser and go to <http://gogs:3000>

Install Steps For First-time Run

If you're running Gogs inside Docker, please read [Guidelines](#) carefully before you change anything in this page!

Database Settings

Gogs requires MySQL, PostgreSQL, SQLite3 or TiDB.

Database Type *

Path *

The file path of SQLite3 or TiDB database.
Please use absolute path when you start as service.

Select SQLite3. Don't
change the path

Application General Settings

Application Name *

Put your organisation name here huge and loud!

Repository Root Path *

All Git remote repositories will be saved to this directory.

Run User *

The user must have access to Repository Root Path and run Gogs.

Domain *

SSH Port *

Port number which your SSH server is using, leave it empty to disable SSH feature.

☐ Use Builtin SSH Server

HTTP Port *

Port number which application will listen on.

Application URL *

This affects HTTP/HTTPS clone URL and somewhere in email.

Log Path *

☐ Enable Console Mode

Di

Set Domain to gogs

Set SSH Port to 10022

Set URL to:
<http://gogs:3000/>

Optional Settings

▸ Email Service Settings

▾ Server and Other Services Settings

☐ Enable Offline Mode

☐ Disable Gravatar Service

☐ Enable Federated Avatars Lookup

☐ Disable Self-registration

☐ Enable Captcha

☐ Enable Require Sign In to View Pages


▸ Admin Account Settings

Expand and ...

... untick Captcha

Install Gogs

After installation you will be taken to sign in. At this point there is no user or admin account so you need to sign up first.

 [Home](#) [Explore](#) [Help](#) [Register](#) [Sign In](#)

Sign In

Welcome! We're glad that you chose Gogs, have fun and take care.

Username or email *

Password *

☐ Remember Me

[Sign In](#) [Forgot password?](#)

[Need an account? Sign up now.](#)

The first user account that you create will automatically get admin privileges.

Sign Up

Username *

Email *

Password *

Re-Type *

[Create New Account](#)

[Already have an account? Sign in now!](#)

When the account is created you can log in and start using Gogs.

Sign In

Username or email *

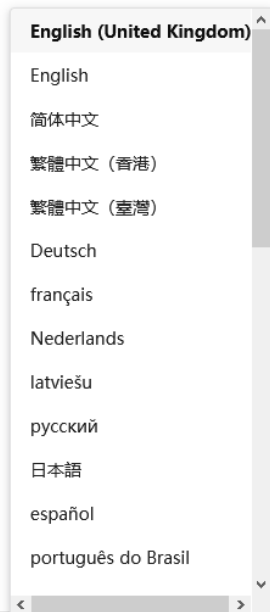
Password *


☐ Remember Me

[Sign In](#) [Forgot password?](#)

[Need an account? Sign up now.](#)

If your UI is not in the language you prefer you can change the language from the menu at the bottom of the page.




Page: 31ms Template: 2ms |  English (United Kingdom) | [Website](#)

Create a new repository

The following shows an example how to create a repository in Gogs and how to push an existing repository to Gogs. When you create a new repository set the name and description of the repository, and leave the rest at their default values. This will create an empty repository.

New Repository

Owner *

 krl

Repository Name *

morse

A good repository name is usually composed of short, memorable and unique keywords.

Visibility

☐ This repository is Private

☐ This repository is Unlisted

Description

Morse sender and Jenkins pipeline

Description of repository. Maximum 512 characters length.

Available characters: 479

.gitignore

Select .gitignore templates

Licence

Select a licence file

Readme ?

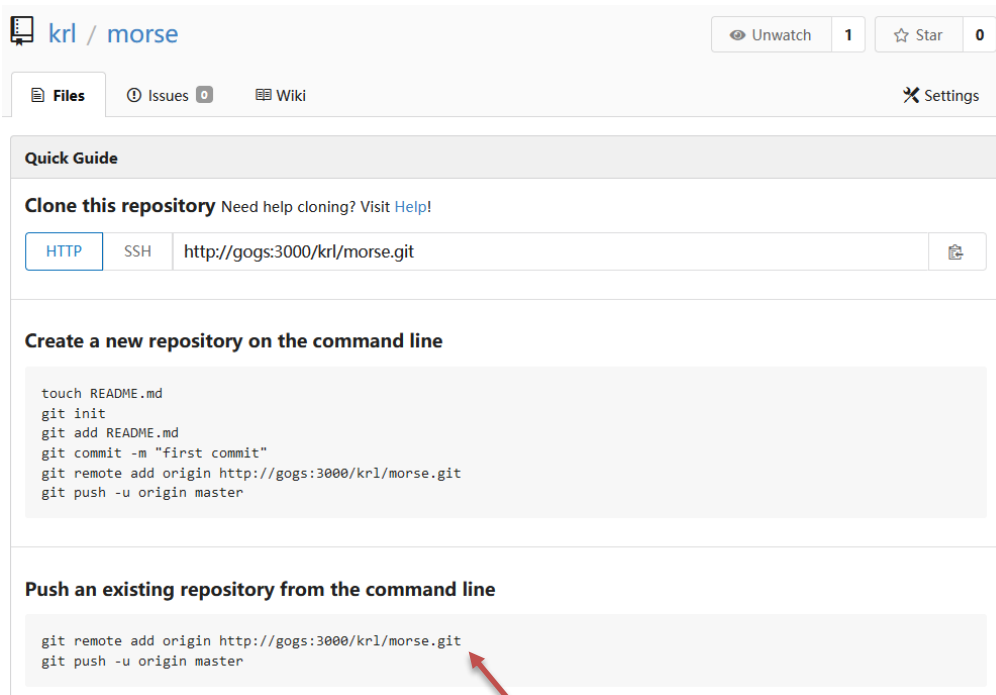
Default

☐ Initialise this repository with selected files and template

Create Repository

Cancel

After creating the repository, you get the quick guide of how to populate the repo with data. We will use the existing repository option with couple of additional actions.



Quick Guide

Clone this repository Need help cloning? Visit [Help!](#)

[HTTP](#) [SSH](#) `http://gogs:3000/krl/morse.git`

Create a new repository on the command line

```
touch README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin http://gogs:3000/krl/morse.git
git push -u origin master
```

Push an existing repository from the command line

```
git remote add origin http://gogs:3000/krl/morse.git
git push -u origin master
```

We start by pulling a repository from Gitlab, or whatever service you use for yours, and then push that repo to Gogs.

Step 1: clone your existing repo

```
$ git clone https://gitlab.metropolia.fi/lansk/morsesender
```

Step 2: cd into the checked out folder and rename the origin. This preserves the link to the original repo in cases where you want to push changes to that repo also.

```
$ cd morsesender
```

```
$ git remote rename origin old-origin
```

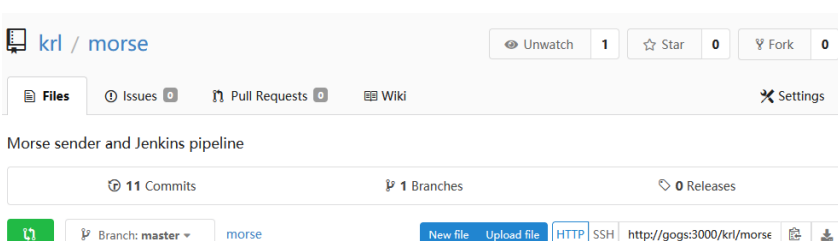
Step 3: set new origin (use steps shown in your repo's quick guide)

```
$ git remote add origin http://gogs:3000/krl/morse.git
```

Step 4: push repo to Gogs (use steps shown in your repo's quick guide)

```
$ git push -u origin master
```

Now you have a repository in Gogs and you are ready start working with Jenkins.



krl / morse [Unwatch](#) [1](#) [Star](#) [0](#) [Fork](#) [0](#)

[Files](#) [Issues](#) [Pull Requests](#) [Wiki](#) [Settings](#)

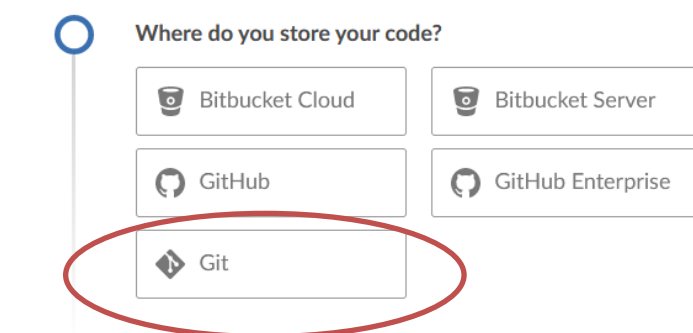
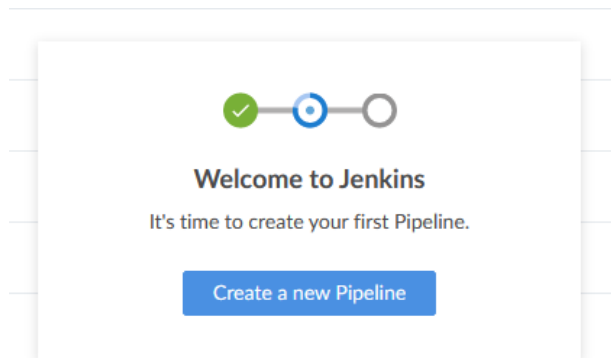
Morse sender and Jenkins pipeline

[11 Commits](#) [1 Branches](#) [0 Releases](#)

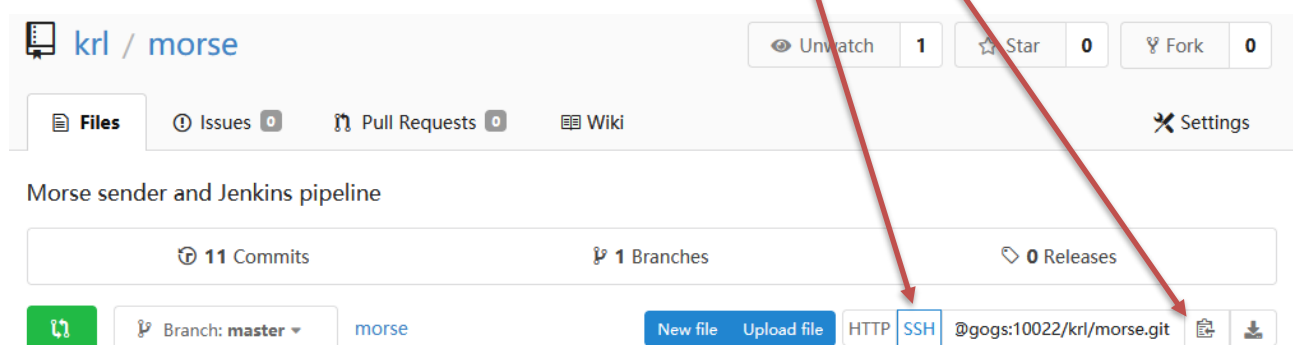
[New file](#) [Upload file](#) [HTTP](#) [SSH](#) `http://gogs:3000/krl/morse` [Push](#)

Create your first Pipeline

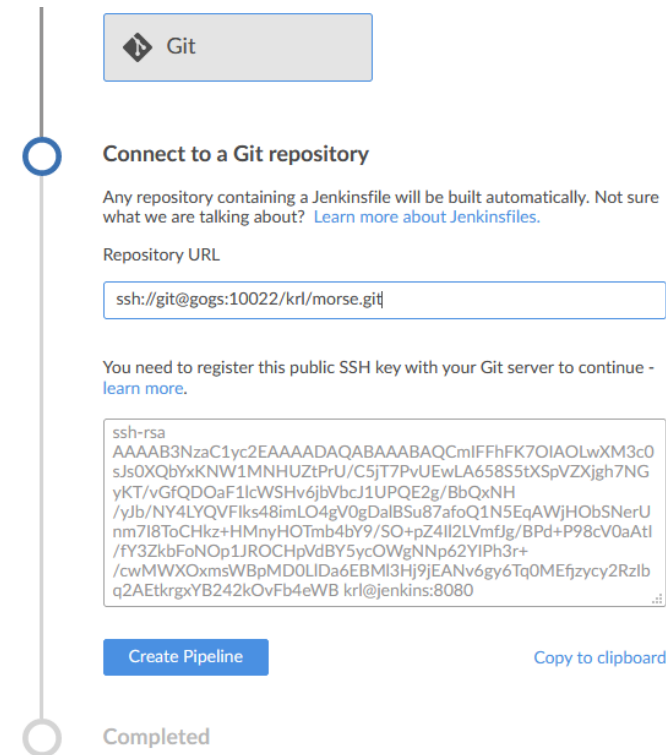
The following shows how to create a Pipeline with Blue Ocean plugin. In the dashboard click “open Ble Ocean”. On the first invocation you will be taken directly to Pipeline creation wizard.



After selecting Git, you need to provide repository URL. Get the URL from Gogs. There are two types of URLs: http and ssh. For Jenkins we use ssh URL because with ssh we can setup public key authentication for easy access to repository. Go to your repository in Gogs. Select SSH and copy the provided URL to clipboard.



When you paste the URL to Repository URL box, Jenkins will automatically create a public key that needs to be installed in Gogs. You need to install the public SSH key to Gogs before you can continue.



The image shows the Jenkins 'Connect to a Git repository' screen. At the top, there's a 'Git' icon. Below it, the title 'Connect to a Git repository' is followed by a paragraph explaining that any repository with a Jenkinsfile will be built automatically. A 'Repository URL' box contains the text 'ssh://git@gogs:10022/krl/morse.git'. Below this, a message states: 'You need to register this public SSH key with your Git server to continue - learn more.' A text box displays a long SSH public key. At the bottom, there are two buttons: 'Create Pipeline' and 'Copy to clipboard'. A progress indicator on the left shows the current step is 'Completed'.

Git

Connect to a Git repository

Any repository containing a Jenkinsfile will be built automatically. Not sure what we are talking about? [Learn more about Jenkinsfiles.](#)

Repository URL

ssh://git@gogs:10022/krl/morse.git

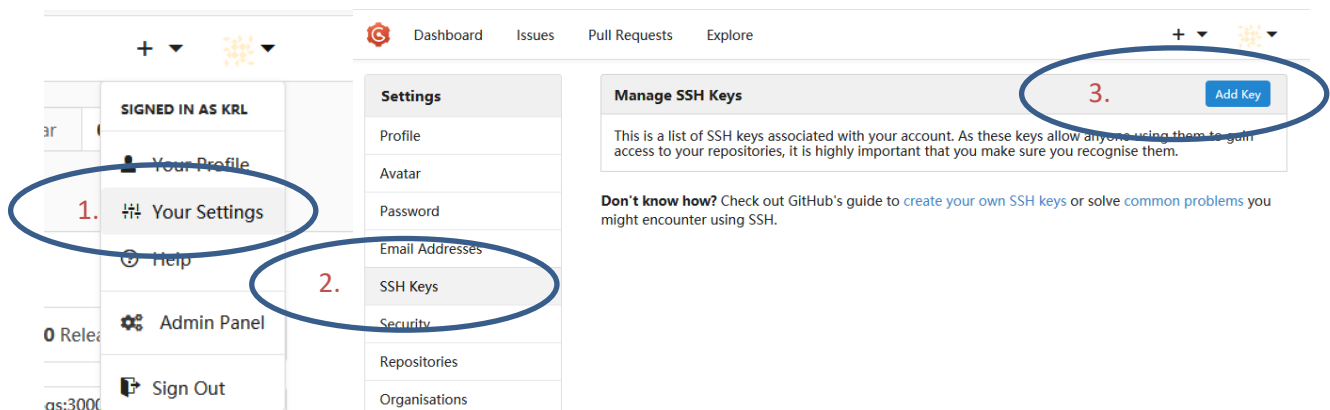
You need to register this public SSH key with your Git server to continue - [learn more.](#)

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCMIFhFK7OIAOLwXM3c0
sJs0XQbYxKNW1MNHUZtPrU/C5jT7PvUEwLA658S5tXSpVZXjgh7NG
yKT/vGfQDOaF1lcWSHv6jbVbcJ1UPQE2g/BbQxNH
/yJb/NY4LYQVFIks48imLO4gV0gDalBSu87afoQ1N5EqAWjHObsNerU
nm7I8ToCHkz+HMnyHOTmb4bY9/SO+pZ4Il2LVmfJg/BPd+P98cV0aAtI
/fY3ZkbFoNOp1JROCHpVdBY5ycOWgNNp62YIPh3r+
/cwMwXOxmsWBpMD0LIDa6EBMI3Hj9JEANv6gy6Tq0MEfjzycy2RzIb
q2AEtKrgxYB242kOvFb4eWB krl@jenkins:8080
```

Create Pipeline Copy to clipboard

Completed

Copy the public key to the clipboard and switch to Gogs. Go to Your settings and select SSH Keys and click Add Key.



The image shows the Gogs user settings page. On the left, a sidebar menu has 'Your Settings' circled in blue and labeled '1.'. In the center, the 'Settings' menu has 'SSH Keys' circled in blue and labeled '2.'. On the right, the 'Manage SSH Keys' section has an 'Add Key' button circled in blue and labeled '3.'. The page also includes a 'Signed in as KRL' header, a 'Dashboard' link, and a 'Don't know how?' section with links to GitHub guides.

SIGNED IN AS KRL

- Your Profile
- 1. Your Settings**
- Help
- Admin Panel
- Sign Out

Settings

- Profile
- Avatar
- Password
- Email Addresses
- 2. SSH Keys**
- Security
- Repositories
- Organisations

Manage SSH Keys

3. Add Key

This is a list of SSH keys associated with your account. As these keys allow anyone using them to gain access to your repositories, it is highly important that you make sure you recognise them.

Don't know how? Check out GitHub's guide to [create your own SSH keys](#) or solve [common problems](#) you might encounter using SSH.

Note: Blue Ocean pipeline editor needs write access to the repository so we need to install the key in user settings.

Name the key and paste the key you copied from Jenkins to Content box.

Manage SSH Keys Add Key

This is a list of SSH keys associated with your account. As these keys allow anyone using them to gain access to your repositories, it is highly important that you make sure you recognise them.

Don't know how? Check out GitHub's guide to [create your own SSH keys](#) or solve [common problems](#) you might encounter using SSH.

Add SSH Key

Key Name

Content

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCFh0Ms/y+aVjm5SighRnlHcHZ
/1O4ThUDeXm5i7LxZSne7jmYlnrI3Lzz7lq5p0EPaVJm2Dpr2Pelw8CxN4Vy8n6yQ7
/K6+OTol5UmXBblnbAVnBLO54A1YyaezXdbo7thi+sPAcv+2+/noijdt
/DDMm5rSVJmmk+bWbsBBYf+2fbJFagLMrVWuez2aHjqGNov5pLp
/7uGINZwiqY3PjpHDVXOzf9AT6eFs2D6D3AmMRHq1cwsioZjOzVSvrOPZDJAnvZCOhIC31tsLpYNUmyk0
/099Av/y4C+46jzQULTO4sC9nxORYYKs+oDNXYaAibM2IXXi7cMbwoAVQQAVz krl@jenkins:8080
```

Add Key


Then click Add Key.

New SSH key 'Jenkins Blue Ocean access' has been added successfully!

Manage SSH Keys Add Key

This is a list of SSH keys associated with your account. As these keys allow anyone using them to gain access to your repositories, it is highly important that you make sure you recognise them.

☐

**Jenkins Blue Ocean access**
SHA256:17mH3OhuXkq7DttGi1NtHTFm8OGxmAMJC1F9HPhrU
Added on Apr 20, 2021 — ⓘ No recent activity

Delete

Don't know how? Check out GitHub's guide to [create your own SSH keys](#) or solve [common problems](#) you might encounter using SSH.


Now you are all set with installing the SSH key.

Switch back to Jenkins and click Create Pipeline.

/CWMVVXUxmsvVBPMUULIDa0EBM13HJYJEANvogyo iquMEtjzcyZKzID
q2AEtkrgxYB242kOvFb4eWB krl@jenkins:8080

Create Pipeline

Copy to clipboard

 Completed

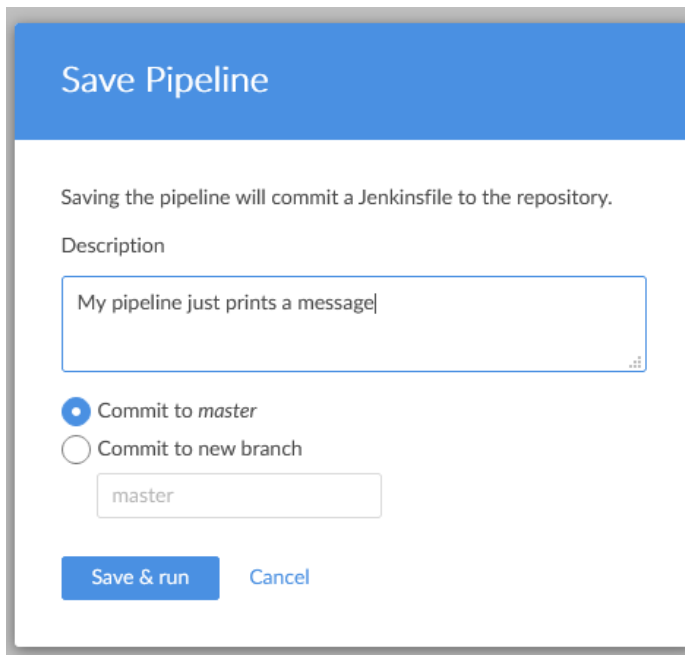
Jenkins verifies that it can access the repository and on success will open the pipeline editor.

The image shows a sequence of five screenshots from the Jenkins Pipeline Editor interface, illustrating the steps to create a new pipeline:

- 1. Click here**: The first screenshot shows the 'Start' node in the pipeline graph. A blue circle highlights the '+' icon next to the 'Start' node, with the text '1. Click here' in red.
- 2. Name the stage**: The second screenshot shows the 'Name your stage' dialog box. A blue circle highlights the input field, with the text '2. Name the stage' in red.
- 3. Add a step**: The third screenshot shows the 'Steps' section. A blue circle highlights the '+ Add step' button, with the text '3. Add a step' in red.
- 4. Choose print message**: The fourth screenshot shows the 'Choose step type' dialog box. A blue circle highlights the 'Print Message' option, with the text '4. Choose print message' in red.
- 5. Enter message**: The fifth screenshot shows the 'Print Message' step configuration. A blue circle highlights the 'Message*' input field, which contains the text 'This is my first pipeline', with the text '5. Enter message' in red.

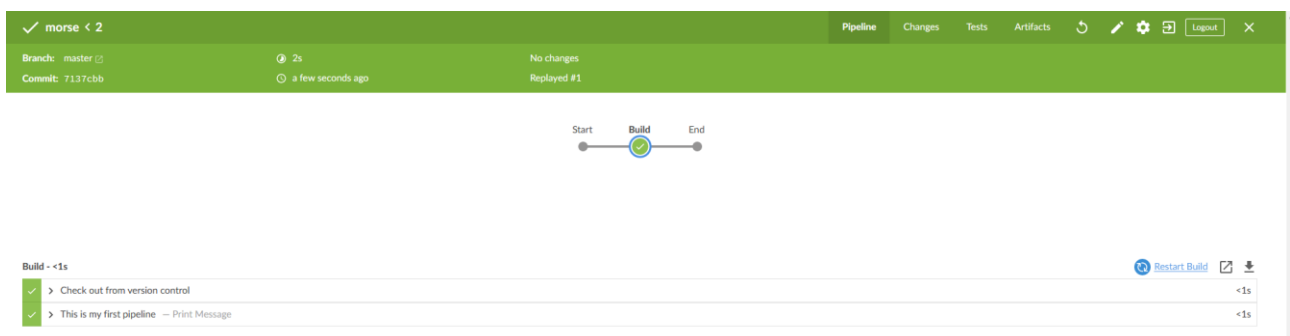
Finally click Save to save your new pipeline.

When you save the Pipeline the created (or modified) Jenkinsfile will be committed to your repository. You will be asked to enter commit message.



The 'Save Pipeline' dialog box has a blue header with the title 'Save Pipeline'. Below the header, a message states: 'Saving the pipeline will commit a Jenkinsfile to the repository.' There is a 'Description' section with a text area containing 'My pipeline just prints a message!'. Below the text area are two radio buttons: 'Commit to master' (selected) and 'Commit to new branch'. Under 'Commit to new branch' is a text input field with 'master' entered. At the bottom are two buttons: 'Save & run' and 'Cancel'.

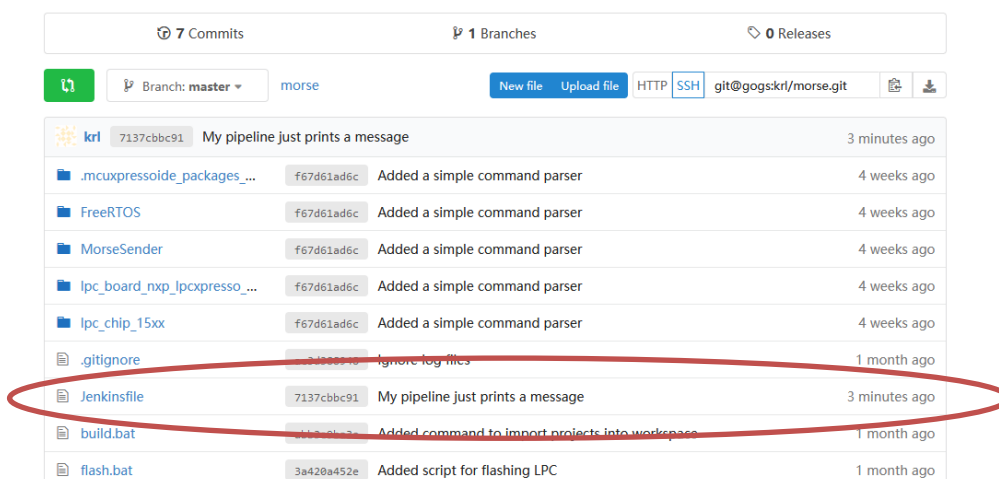
When you press Save & Run your Jenkinsfile will be saved and then the pipeline is run. Jenkins automatically checks out the repository and runs the Pipeline.



The Jenkins Pipeline view shows a green header bar with the pipeline name 'morse < 2' and various icons. Below the header, it shows 'Branch: master' and 'Commit: 7137cbb'. A progress bar indicates the 'Build' step is completed. Below the progress bar, a list of build steps is shown: 'Check out from version control' and 'This is my first pipeline - Print Message'. A 'Restart Build' button is visible on the right.

As you can see a new file was added to the repository.

Morse sender and Jenkins pipeline



The GitHub repository view for 'morse' shows 7 commits, 1 branch, and 0 releases. The commit list includes files like '.mcuxpressoide_packages_...', 'FreeRTOS', 'MorseSender', 'lpc_board_nxp_lpcpresso_...', 'lpc_chip_15xx', '.gitignore', 'Jenkinsfile', 'build.bat', and 'flash.bat'. The 'Jenkinsfile' commit is circled in red.

File	Commit Hash	Message	Time
.mcuxpressoide_packages_...	f67d61ad6c	Added a simple command parser	4 weeks ago
FreeRTOS	f67d61ad6c	Added a simple command parser	4 weeks ago
MorseSender	f67d61ad6c	Added a simple command parser	4 weeks ago
lpc_board_nxp_lpcpresso_...	f67d61ad6c	Added a simple command parser	4 weeks ago
lpc_chip_15xx	f67d61ad6c	Added a simple command parser	4 weeks ago
.gitignore	f67d61ad6c	Ignore log files	1 month ago
Jenkinsfile	7137cbb91	My pipeline just prints a message	3 minutes ago
build.bat	f67d61ad6c	Added command to import projects into workspace	1 month ago
flash.bat	3a420a452e	Added script for flashing LPC	1 month ago

Take a look at the Jenkinsfile



Naturally we can also edit the file manually.

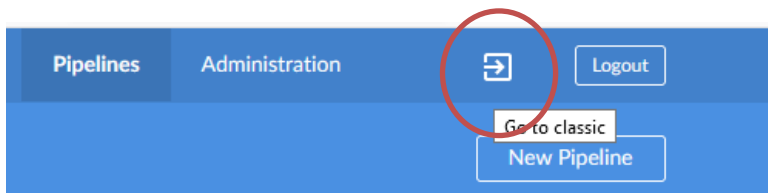
Pull Jenkinsfile from Gogs to local repository:

```
$ git pull
```

```
remote: Enumerating objects: 4, done.  
remote: Counting objects: 100% (4/4), done.  
remote: Compressing objects: 100% (3/3), done.  
remote: Total 3 (delta 1), reused 0 (delta 0)  
Unpacking objects: 100% (3/3), 360 bytes | 22.00 KiB/s, done.  
From http://gogs:3000/krl/morse  
    f67d61a..7137cbb master    -> origin/master  
Updating f67d61a..7137cbb  
Fast-forward  
jenkinsfile | 11 ++++++++  
1 file changed, 11 insertions(+)  
create mode 100644 jenkinsfile
```

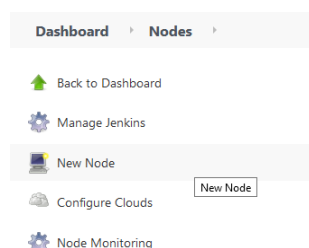
Create a node

If you are in Blue Ocean go first back to "classic" Jenkins.



Go to Dashboard → Manage Jenkins → Manage Nodes and Clouds

Click New node



Name the node and select Permanent agent.

Node name

My local build machine

☒ **Permanent Agent**

Adds a plain, permanent agent to Jenkins. This is called "permanent agent". If you are adding a physical computer, virtual machines manager, etc.

☐ **Copy Existing Node**

Copy from

OK

After pressing OK, you will get to configure the agent.

Name

My local build machine

Description

Windows10 with MCUXpresso

of executors

1

Remote root directory

C:\Jenkins\

Labels

windows

Usage

Only build jobs with label expressions matching this node

Launch method

Launch agent by connecting it to the master

☐ Disable WorkDir

Custom WorkDir path

Internal data directory

remoting

☐ Fail if workspace is missing

☐ Use WebSocket

Availability

Keep this agent online as much as possible

Node Properties

☐ Disable deferred wipeout on this node

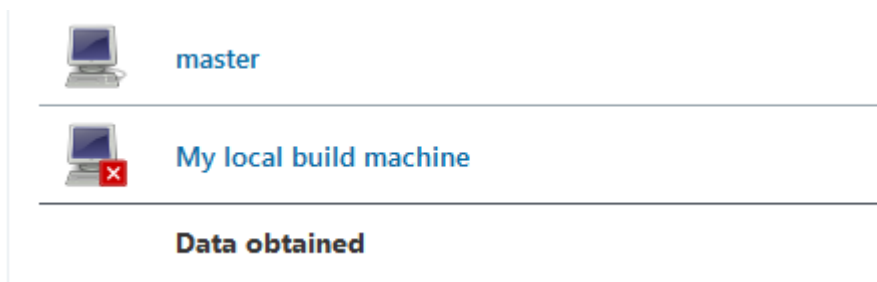
☐ Environment variables

☐ Tool Locations

Save

Important
settings!

After saving the node, you will return node view and see that your agent is offline.




When you click your new node, you get to start the node. See the screen capture below.



Agent My local build machine (Windows10 with MCUXpresso)

Connect agent to Jenkins one of these ways:

-  **Launch** Launch agent from browser
- Run from agent command line:

```
java -jar agent.jar -jnlpUrl http://jenkins:8080/computer/My%20local%20build%20machine/jenkins-agent.jnlp -secret "C:\Jenkins\"
```

Run from agent command line, with the secret stored in a file:

```
echo 9993a6b519ade95433367dbf4cad539be064401244f3f8af8390b4c0c7378fd > secret-file  
java -jar agent.jar -jnlpUrl http://jenkins:8080/computer/My%20local%20build%20machine/jenkins-agent.jnlp -secret
```

Labels

windows

Projects tied to My local build machine

None

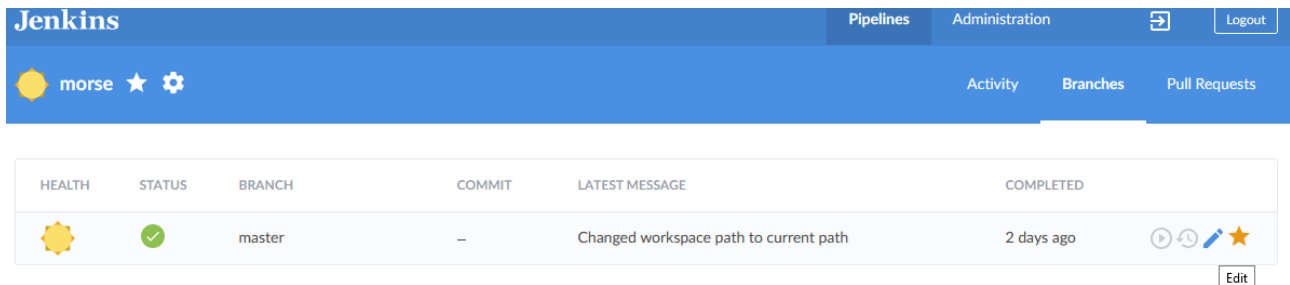
Pressing launch button download a jnlp-file. Firefox refuses to launch jnlp-files automatically so I saved the jnlp-file. Then I launched the jnlp to start the Agent. If you close the agent your node will be offline. When you click on the offline node you get this same start information again. If you save this information as jnlp-file or as a script you can easily start the agent without having to use Jenkins GUI.

In the newer versions of Jenkins the launch button is not shown. You have two options:

1. Copy [the link from the command line](#) and bookmark it in your browser. Then you can start the agent from your browser.
2. Click on **agent.jar** and save it to a directory, for example c:\Jenkins. Create a batch file with the whole command line and create a shortcut to the batch file on the desktop or what ever your favourite place is.

How to edit a Pipeline

1. Open blue ocean
2. Select Branches (should be default view)
3. Move cursor to end of line and click on pen icon to edit the pipeline.



How to automate build

The example project contains two batch files: build.bat and flash.bat.

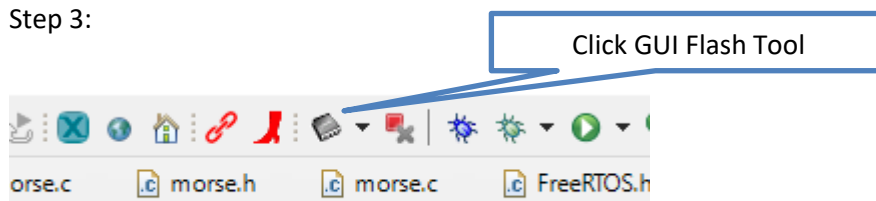
Build.bat performs a headless (no-GUI) build of the morse sender project. Note that you need to edit the batch file to match your current setup (install directory). To test the headless build without Jenkins do a fresh checkout to an empty test directory and run the batch in the checked-out directory.

Flash.bat contains a script to flash the built image to LPC-board. This is also installation specific. You can easily get script for your installation from the gui.

Step 1. Connect LPC to you computer

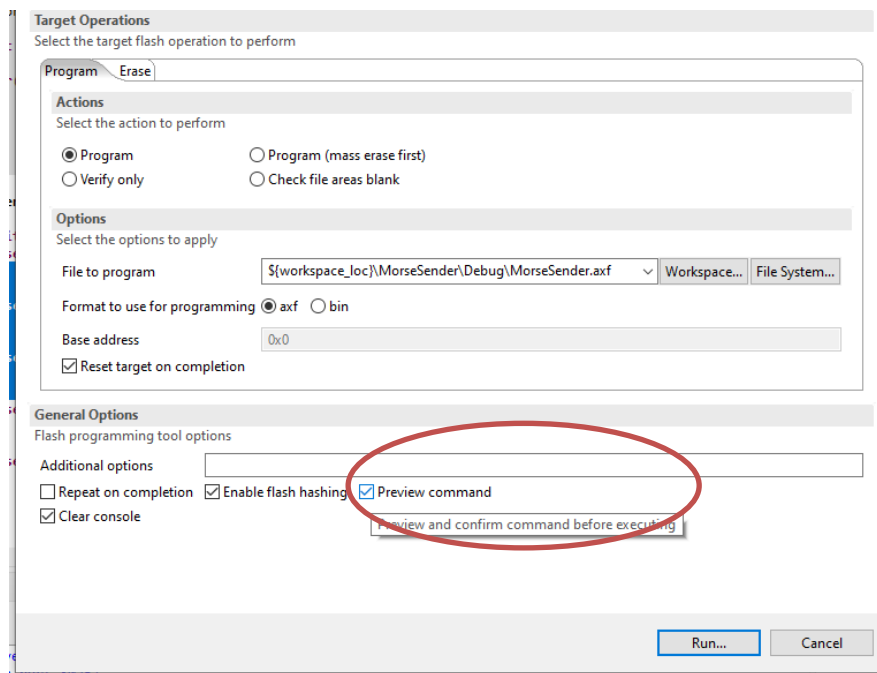
Step 2: Build morsesender in GUI

Step 3:



1ue):

Step 4: Check "Preview command" and click "Run..."



Step 5: Select the type of script you want and copy the script to flash.bat.

