Jenkins File

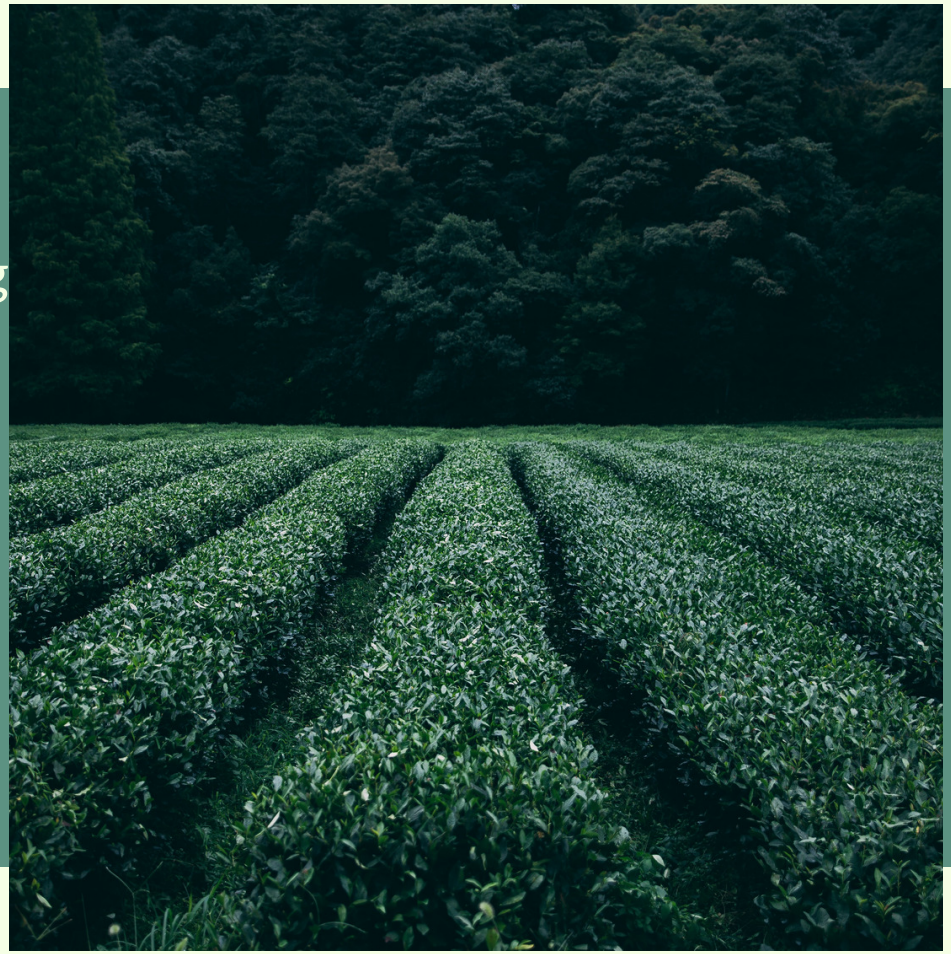
JENKINS

BY ALOK SRIVASTAVA - NETWORK NUTS

Network Nuts
Love Learn Linux

JENKINSFILE

A pipeline in Jenkins is defined using a script called the Jenkinsfile. This provides more automation, allows your pipeline to be treated like all other application code, and can be stored in version control.



The pipeline syntax can be presented in two forms: **declarative** and **scripted**.

The declarative syntax is a simple and opinionated way of writing your pipeline.

The scripted pipeline is built with Groovy and is generally a more flexible and expressive way of creating your pipelines. When choosing which model to use, it all depends on your requirements.

The declarative model works with simple pipelines and lacks most of the flexibility offered by the scripted model.

We will be working with the scripted pipeline

The scripted pipeline has some special directives that perform different functions.

Directive	Explanation node
`node`	This defines where the job is going to be run. We will explore more about this in the next chapter as we cover setting up master-slave relationships on Jenkins.
`dir`	This directive defines what directory/folder to run the following directives on.
`stage`	This defines the stage of your pipeline, for example, what task it's running.
`git`	This points to the remote repository where you pull the changes from.
`sh`	This defines the shell script to run on a UNIX-based environment. On a Windows environment, we would use the `bat` directive instead.
`def`	As mentioned previously, the pipeline is written in Groovy; thus, we can define functions to perform different actions. In this case, we defined a `printMessage` function, which prints out different messages at the start and end of our pipeline.

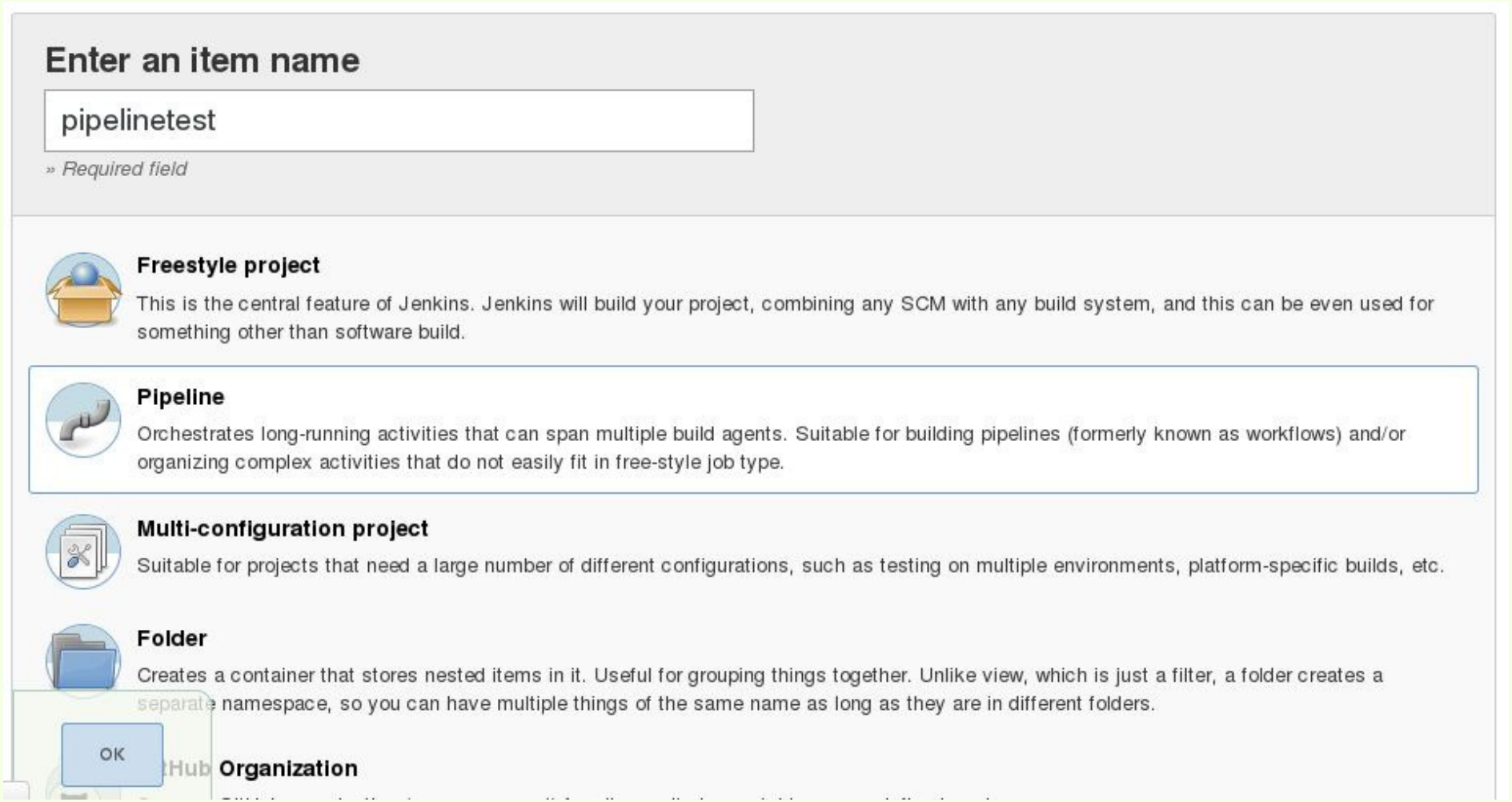
Creating the Pipeline.

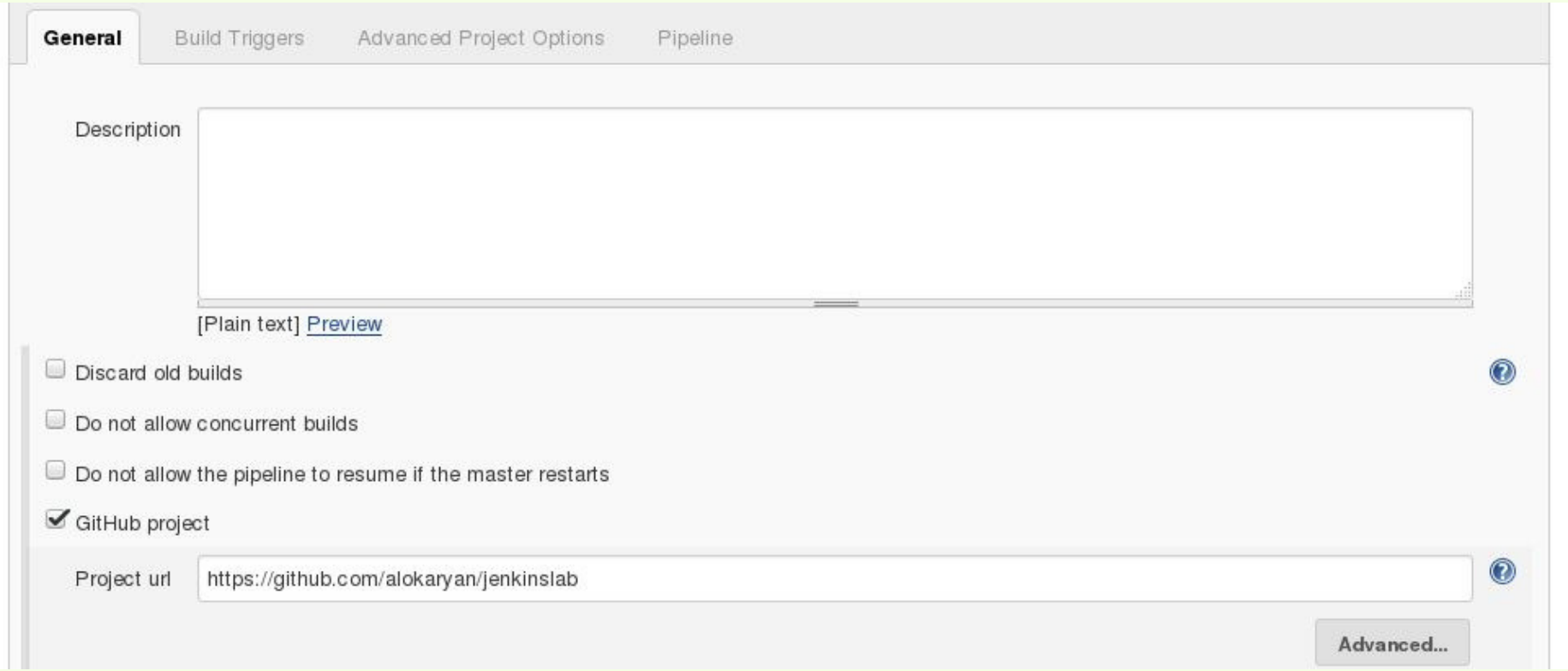
Go to the Jenkins dashboard and select New Item.

Enter an appropriate name for the project and select Pipeline for the project type.

In the project configuration, under the General tab,

select GitHub project and enter the appropriate URL.





Under the **Build Triggers** section, select the **GitHub hook trigger for GITScm polling**, which will help us automatically trigger builds on our pipeline whenever a commit is pushed to GitHub. This configuration will only work with a **hosted Jenkins server** and configured GitHub webhook on the repository.

The final configuration of the project is creating our pipeline.

Your final pipeline script configuration should look like this

Pipeline script

Script

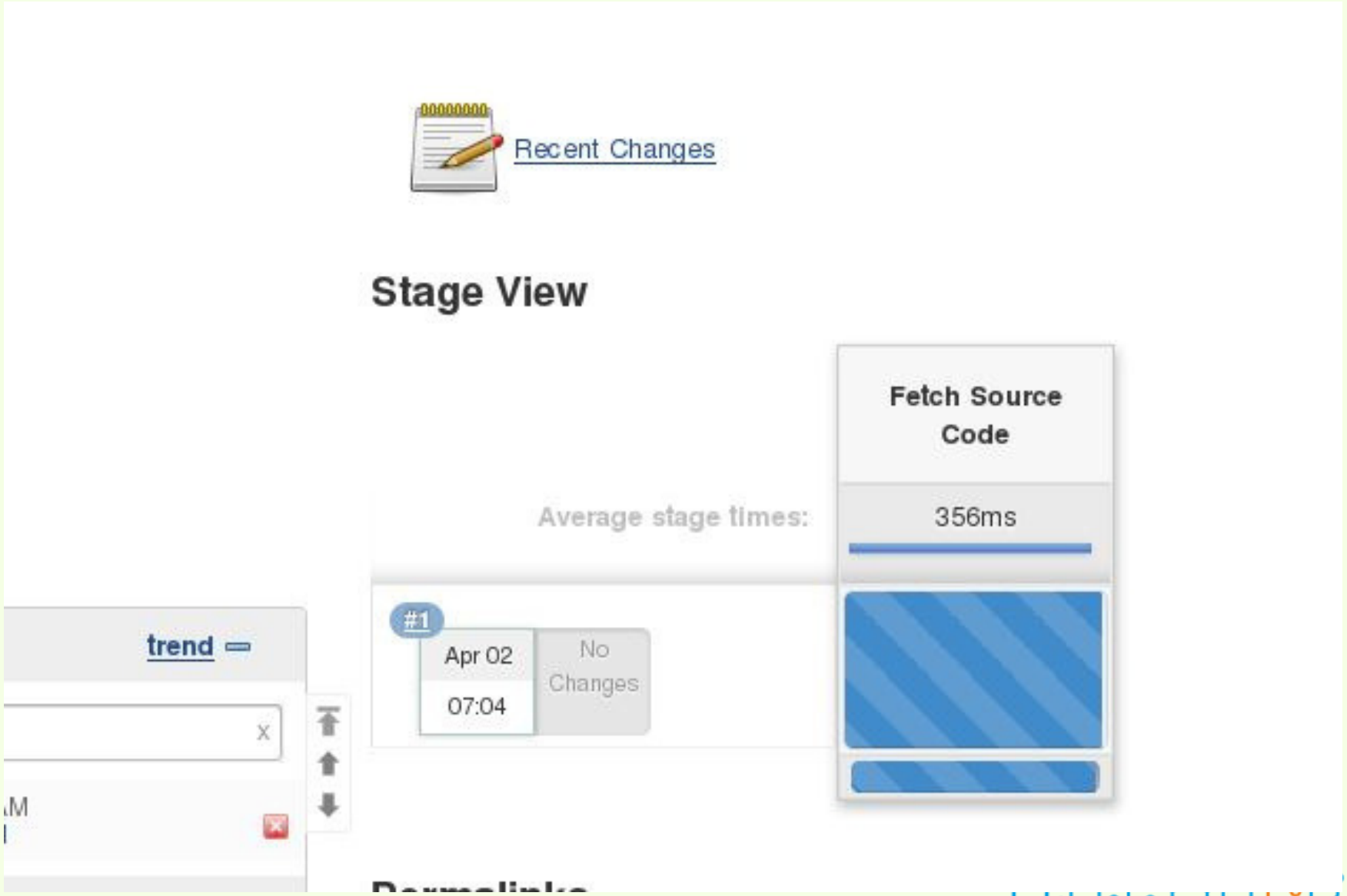
```
1  node('master') {
2      stage("Fetch Source Code") {
3          git "https://github.com/alokaryan/jenkinslab"
4      }
5          dir('jenkinslab') {
6          printMessage('Running Pipeline')
7              stage("Testing") {
8              sh 'python echo.py'
9          }
10             printMessage('Job Completed')
11     }
12 }
13
14 def printMessage(message){
15     echo "${message}"
16 }
17
```

☑ Use Groovy Sandbox

Press **Apply to save** the configuration so far before Jenkins logs you out, which may result in some data loss.

Select Save to persist your configuration and redirect you back to the project dashboard.

Select Build Now on the left-hand menu to build your project. Since we have no commits to push, we will manually trigger a build.


Recent Changes
Stage View
Fetch Source Code
Average stage times:
356ms
#1
Apr 02
07:04
No Changes
trend

```
Running on Jenkins in /var/lib/jenkins/workspace/pipelinetest
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Fetch Source Code)
[Pipeline] git
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/alokaryan/jenkinslab
 > git init /var/lib/jenkins/workspace/pipelinetest # timeout=10
Fetching upstream changes from https://github.com/alokaryan/jenkinslab
 > git --version # timeout=10
 > git fetch --tags --progress https://github.com/alokaryan/jenkinslab +refs/heads/*:re
 > git config remote.origin.url https://github.com/alokaryan/jenkinslab # timeout=10
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=1
 > git config remote.origin.url https://github.com/alokaryan/jenkinslab # timeout=10
Fetching upstream changes from https://github.com/alokaryan/jenkinslab
 > git fetch --tags --progress https://github.com/alokaryan/jenkinslab +refs/heads/*:re
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 68488740778c03f2829c5f24304d0c2df5c72617 (refs/remotes/origin/mas
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 68488740778c03f2829c5f24304d0c2df5c72617
 > git branch -a -v --no-abbrev # timeout=10
 > git checkout -b master 68488740778c03f2829c5f24304d0c2df5c72617
Commit message: "Merge pull request #2 from alokaryan/test"
First time build. Skipping changelog.
[Pipeline] }
```


installing
BLUE OCEAN

Jenkins has a new way of visualizing pipelines, called Blue Ocean.

On the Jenkins home dashboard, go to Manage Jenkins -> Manage Plugins. Under the Available tab, search for Blue Ocean.

Jenkins

- New Item
- People
- Build History
- Manage Jenkins
- My Views
- Lockable Resources
- Credentials
- Open Blue Ocean
- New View

Build Queue

No builds in the queue.

Network Nuts

All +

S	W	Name ↓	Last Success
		ONE	2 days 3 hr - #1
		pipelinetest	N/A
		projectparameter	2 days 4 hr - #2
		testgit	2 days 5 hr - #2
		TWO	2 days 3 hr - #1

Icon: S M L