V2 Lab 2 Steps 2B, 2C & 2D

Like | Updated 16 June 2017 by Katamneni, Krishna Sravya | Tags: *None*

Atom	`Anything in this box needs to be edited in Atom Text`

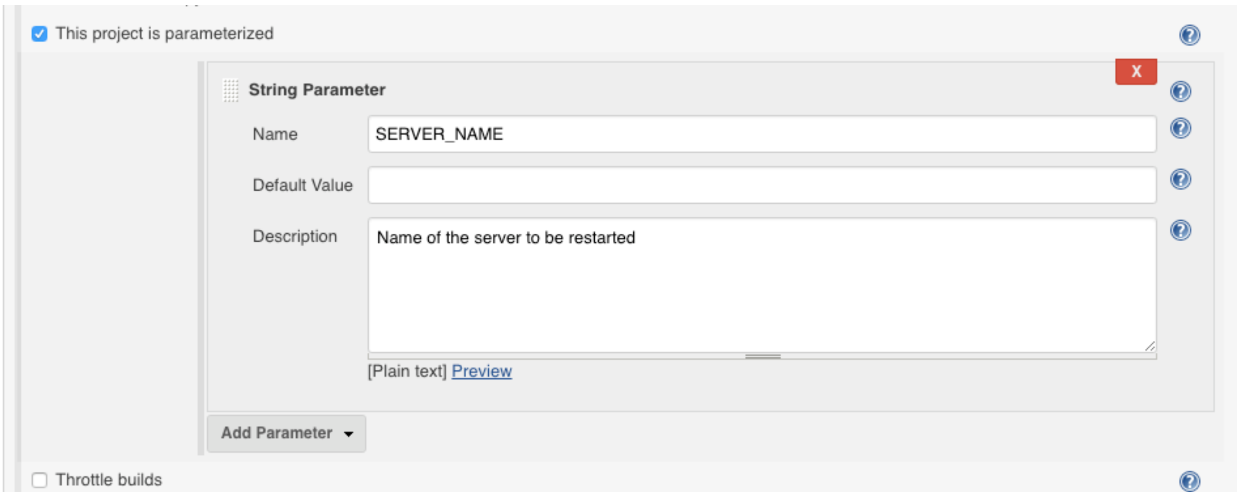
Terminal	`This box contains lists of commands that should executed in order on the Terminal`

(B) Create helper jobs such as server restarts

The objective of this lab is to create some -ops jobs for administrating servers. Tasks such as stopping / starting servers are useful candidates for automation. Below you will create three jobs, one for stopping, one for restarting the app servers and one for restarting the database container

– On Jenkins; create a *New Item* > *Freestyle project* on Jenkins giving it a sensible name such as `generic-server-stop`

– Mark this job as parameterized and enter SERVER_NAME in the name field and give it a description. This variable will be exposed to the job when it runs and it will be used to decide which container should be used for log collection.


☑ This project is parameterized

String Parameter

Name: SERVER_NAME

Default Value:

Description: Name of the server to be restarted

[Plain text] Preview

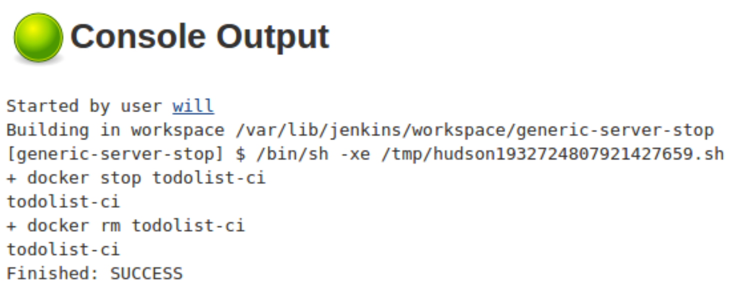
Add Parameter

☐ Throttle builds

– Move to the *Build* tab and hit the drop down *Add build step* > *execute shell* step. In the box add the following to get the logs from the container. We are adding the data and saving the file in the workspace with the app name and log prefixed on the file.

```
docker stop ${SERVER_NAME}  && docker rm ${SERVER_NAME}
```

– *Save* the job configuration and run the job by passing in the name of one of the servers eg `todolist-ci`



Console Output

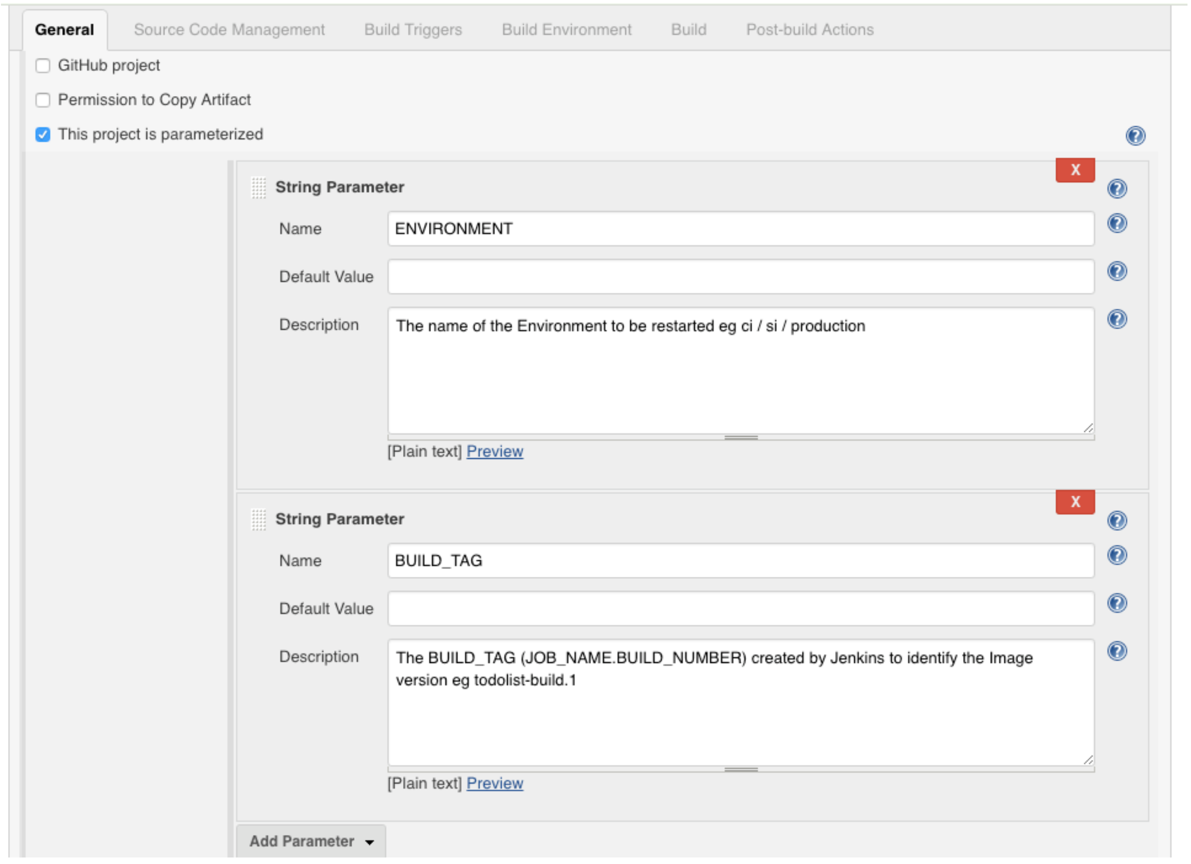
```
Started by user will
Building in workspace /var/lib/jenkins/workspace/generic-server-stop
[generic-server-stop] $ /bin/sh -xe /tmp/hudson1932724807921427659.sh
+ docker stop todolist-ci
todolist-ci
+ docker rm todolist-ci
todolist-ci
Finished: SUCCESS
```

Note: this will fail if the containers are already in a stopped state

Now that the containers are stopped, create a new job for restarting them so they can be powered back up.

– On Jenkins; create a *New Item* > *Freestyle project* on Jenkins giving it a sensible name such as `generic-server-restart`

– Mark this job as parameterized and enter `ENVIRONMENT` in the name field and give it a description. This variable will be exposed to the job when it runs and it will be used to decide which container should be used for server restart. Add another field `BUILD_TAG` too, this will be used to identify the container that Jenkins should use.



☐ GitHub project
☐ Permission to Copy Artifact
☑ This project is parameterized

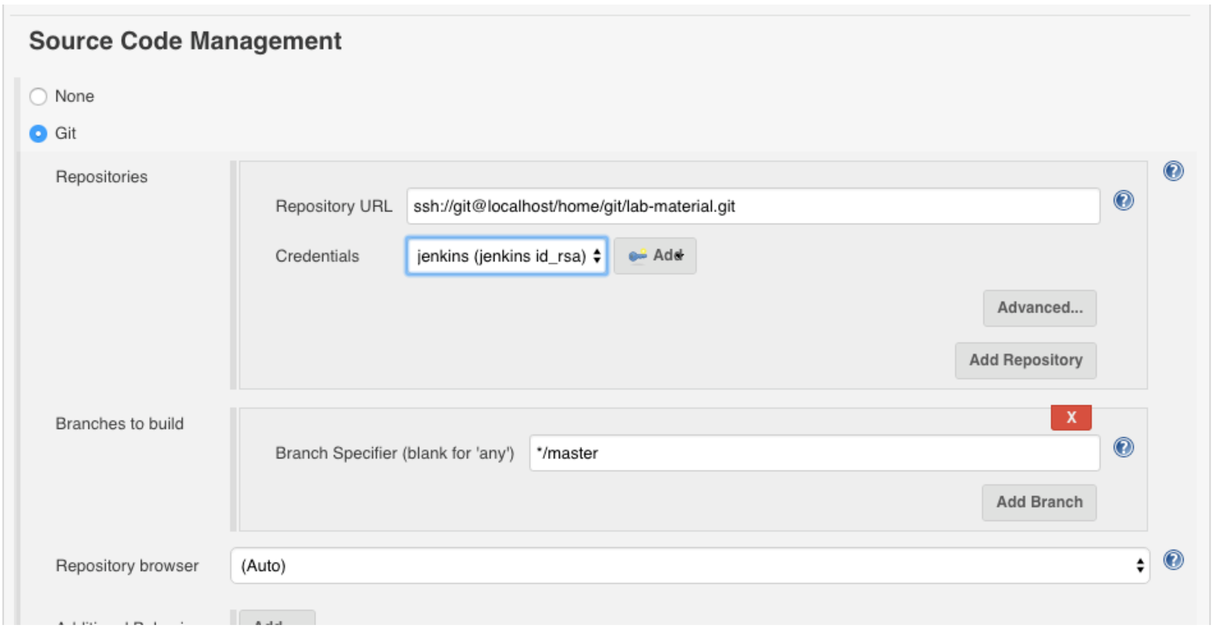
String Parameter

Name	ENVIRONMENT
Default Value	
Description	The name of the Environment to be restarted eg ci / si / production

String Parameter

Name	BUILD_TAG
Default Value	
Description	The BUILD_TAG (JOB_NAME.BUILD_NUMBER) created by Jenkins to identify the Image version eg todolist-build.1

– To keep things simple, a bash script has been written for administering the containers and making sure the correct dependencies exist for each environment. This is in the `/share/lab-material/scripts/jenkins-restart-app.sh` file. To access it in our job, Jenkins must check it out. Add git and then the remote `ssh://git@localhost/home/git/lab-material.git` on the Source Code Management tab. Select the `jenkins (jenkins id_rsa)` credentials. Branch can be left at `*/master`


Source Code Management

None
[✓ Git]

Repositories
Repository URL: ssh://git@localhost/home/git/lab-material.git
Credentials: jenkins (jenkins id_rsa)

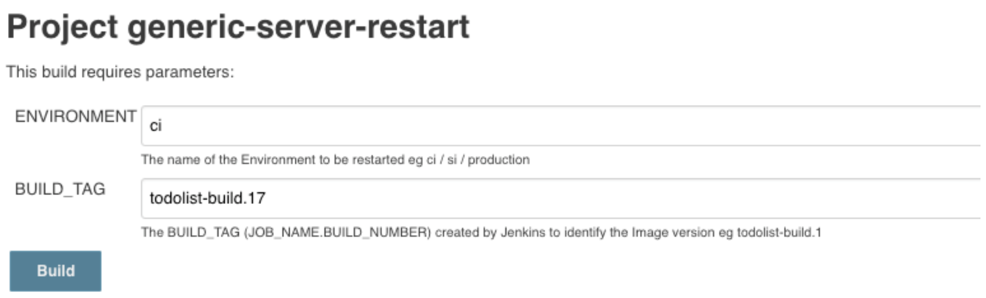
Branches to build
Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

– Move to the *Build* tab and hit the drop down *Add build step > execute shell* step. In the box add the following to get the server passed in to the restart script.

```
./scripts/jenkins-restart-app.sh ${ENVIRONMENT} ${BUILD_TAG}
```

– *Save* the job configuration and run the job by passing in the name of one of the builds and the environment eg `ci` and `todolist-build.17`


Project generic-server-restart

This build requires parameters:

ENVIRONMENT: ci
The name of the Environment to be restarted eg ci / si / production

BUILD_TAG: todolist-build.17
The BUILD_TAG (JOB_NAME.BUILD_NUMBER) created by Jenkins to identify the Image version eg todolist-build.1

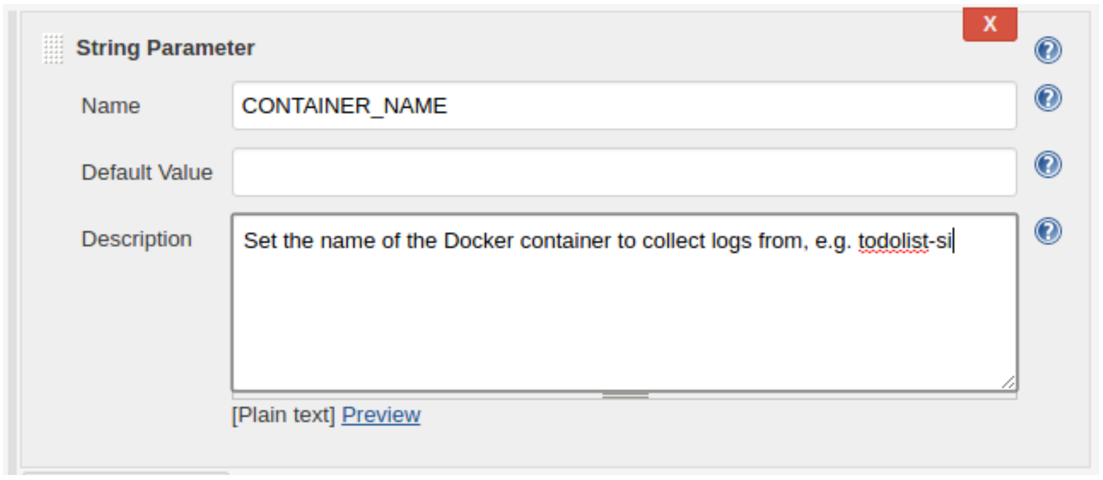
HINT: open the terminal and type `docker ps` to view running containers that can be stopped. `docker ps -a` will show you both stopped and running containers. The `NAMES` will be suffixed with the environment name and the `IMAGE` will contain the valid build tag after the colon.

(C) Create helper jobs such as log file collection

The objective of this lab is to create some automation helper jobs to provide people the ability to access information which may be useful to them, such as server logs for audits. To do this, we will create a parameterized job that will take the name of the thing we want to get logs on.

– On Jenkins; create a *New Item > Freestyle project* on Jenkins giving it a sensible name such as `generic-server-logs`

– Mark this job as parameterized and enter CONTAINER_NAME in the name field and give it a description. This variable will be exposed to the job when it runs and it will be used to decide which container should be used for log collection.



String Parameter

Name: CONTAINER_NAME

Default Value:

Description: Set the name of the Docker container to collect logs from, e.g. todolist-si

[Plain text] Preview

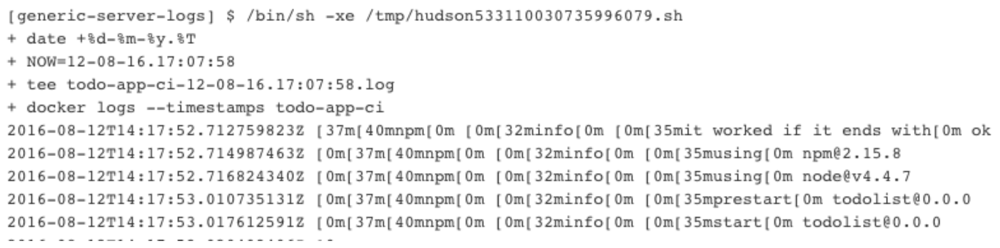
– Enter a sensible description for the Job.

– Move to the *Build* tab and add execute shell step. In the box add the following to get the logs from the container. We are adding the data and saving the file in the workspace with the app name and log prefixed on the file.

```
NOW=$(date +"%d-%m-%y.%T")
docker logs --timestamps ${CONTAINER_NAME} | tee ${CONTAINER_NAME}-${NOW}.log
```

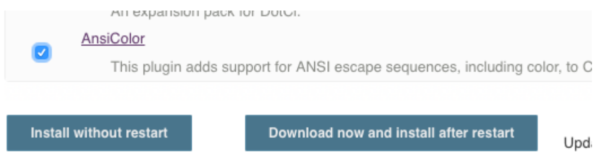
– Save and build the job passing in a param such as `todolist-ci`

– The console output should show the logs but it will be all messy and hard to read. This is because the output from the node app is in colour and Jenkins doesn't know how to understand it.



```
[generic-server-logs] $ /bin/sh -xe /tmp/hudson5331100307359960 79.sh
+ date +%d-%m-%y.%T
+ NOW=12-08-16.17:07:58
+ tee todo-app-ci-12-08-16.17:07:58.log
+ docker logs --timestamps todo-app-ci
2016-08-12T14:17:52.712759823Z [37m[40mnpm[0m [0m[32minfo[0m [0m[35mit worked if it ends with[0m ok
2016-08-12T14:17:52.714987463Z [0m[37m[40mnpm[0m [0m[32minfo[0m [0m[35musing[0m npm@2.15.8
2016-08-12T14:17:52.716824340Z [0m[37m[40mnpm[0m [0m[32minfo[0m [0m[35musing[0m node@v4.4.7
2016-08-12T14:17:53.010735131Z [0m[37m[40mnpm[0m [0m[32minfo[0m [0m[35mprestart[0m todolist@0.0.0
2016-08-12T14:17:53.017612591Z [0m[37m[40mnpm[0m [0m[32minfo[0m [0m[35mstart[0m todolist@0.0.0
```

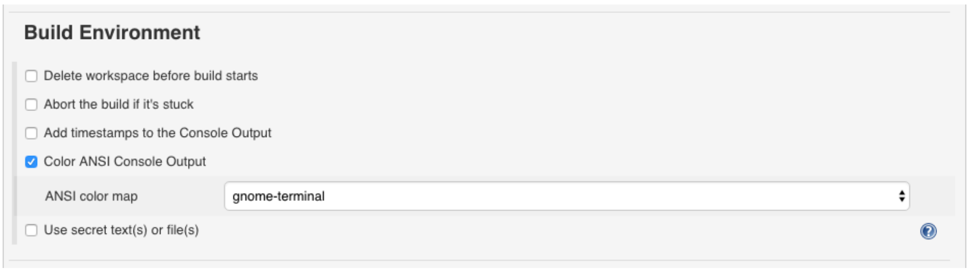
– To fix the colour output on the job, install the *AnsiColor Plugin* in Jenkins.



AnsiColor

This plugin adds support for ANSI escape sequences, including color, to C

– Go back to the job configuration and in the *Build Environment* tab, tick the `ANSI Console Output` and set it to `gnome-terminal`



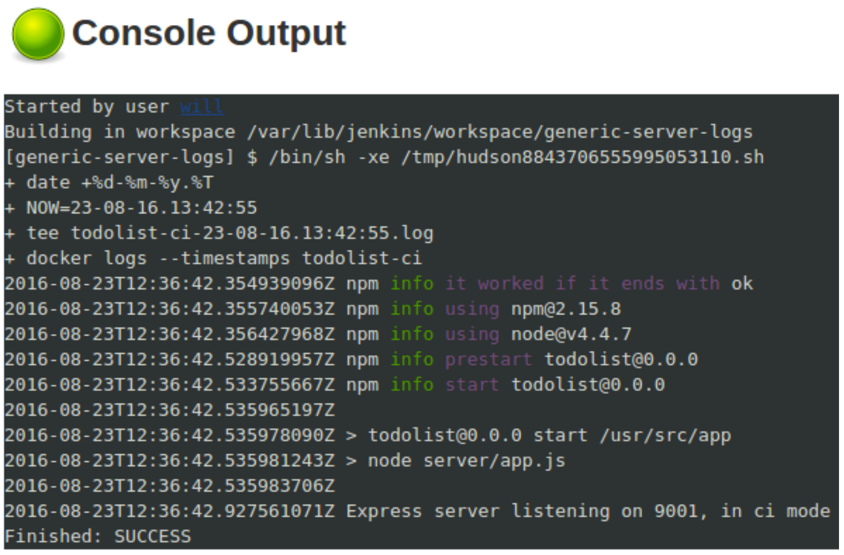
Build Environment

- [ ] Delete workspace before build starts
- [ ] Abort the build if it's stuck
- [ ] Add timestamps to the Console Output
- [x] Color ANSI Console Output

ANSI color map: gnome-terminal

- [ ] Use secret text(s) or file(s)

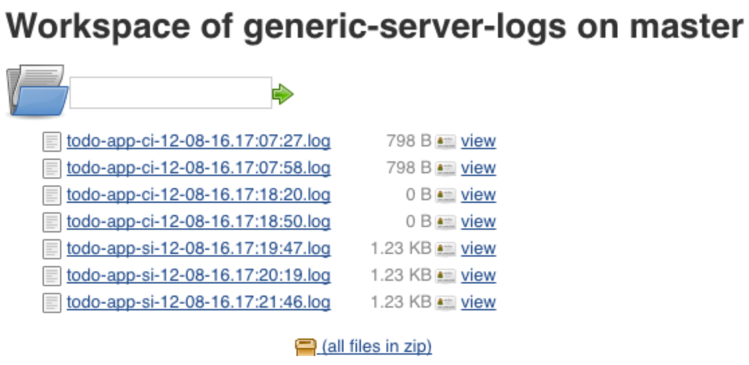
– Re run the job and see the output well formatted:



Console Output

```
Started by user will
Building in workspace /var/lib/jenkins/workspace/generic-server-logs
[generic-server-logs] $ /bin/sh -xe /tmp/hudson8843706555995053110.sh
+ date +%d-%m-%y.%T
+ NOW=23-08-16.13:42:55
+ tee todolist-ci-23-08-16.13:42:55.log
+ docker logs --timestamps todolist-ci
2016-08-23T12:36:42.354939096Z npm info it worked if it ends with ok
2016-08-23T12:36:42.355740053Z npm info using npm@2.15.8
2016-08-23T12:36:42.356427968Z npm info using node@v4.4.7
2016-08-23T12:36:42.528919957Z npm info prestart todolist@0.0.0
2016-08-23T12:36:42.533755667Z npm info start todolist@0.0.0
2016-08-23T12:36:42.535965197Z
2016-08-23T12:36:42.535978090Z > todolist@0.0.0 start /usr/src/app
2016-08-23T12:36:42.535981243Z > node server/app.js
2016-08-23T12:36:42.535983706Z
2016-08-23T12:36:42.927561071Z Express server listening on 9001, in ci mode
Finished: SUCCESS
```

- The history of previously gotten logs can then be found on the *Workspace*.



Workspace of generic-server-logs on master

todo-app-ci-12-08-16.17:07:27.log	798 B	view
todo-app-ci-12-08-16.17:07:58.log	798 B	view
todo-app-ci-12-08-16.17:18:20.log	0 B	view
todo-app-ci-12-08-16.17:18:50.log	0 B	view
todo-app-si-12-08-16.17:19:47.log	1.23 KB	view
todo-app-si-12-08-16.17:20:19.log	1.23 KB	view
todo-app-si-12-08-16.17:21:46.log	1.23 KB	view

(all files in zip)

(D) Homework Tasks

1. Create a List View on jenkins for for all of the helper jobs. Hint for Regex `generic-.*`
2. Add the *rebuild* plugin to make it easier to restart the job with last runs parameters.
3. Add ANSI Output to other jobs for convenience
4. Create a job that collects the VMs logs.

Advanced Extension Task

The objective of this lab is to use docker-compose to create a environment definition file with two node servers, a mongo db and a webserver in front to mimic a more 'prod-like' environment

-

Comments

There are no comments.