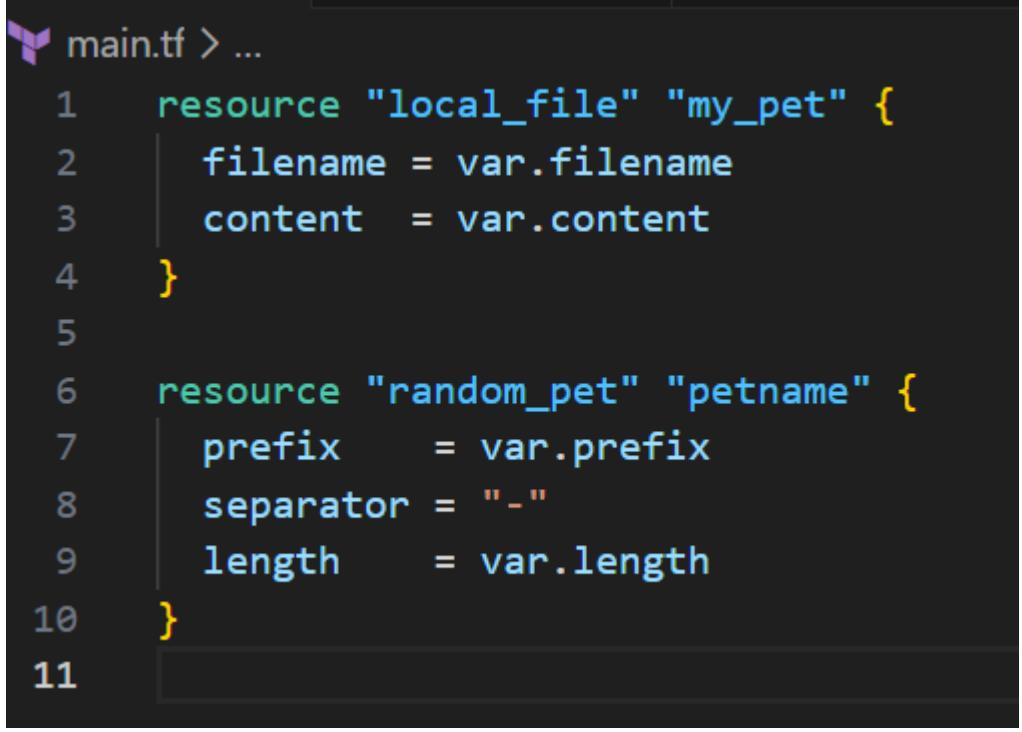


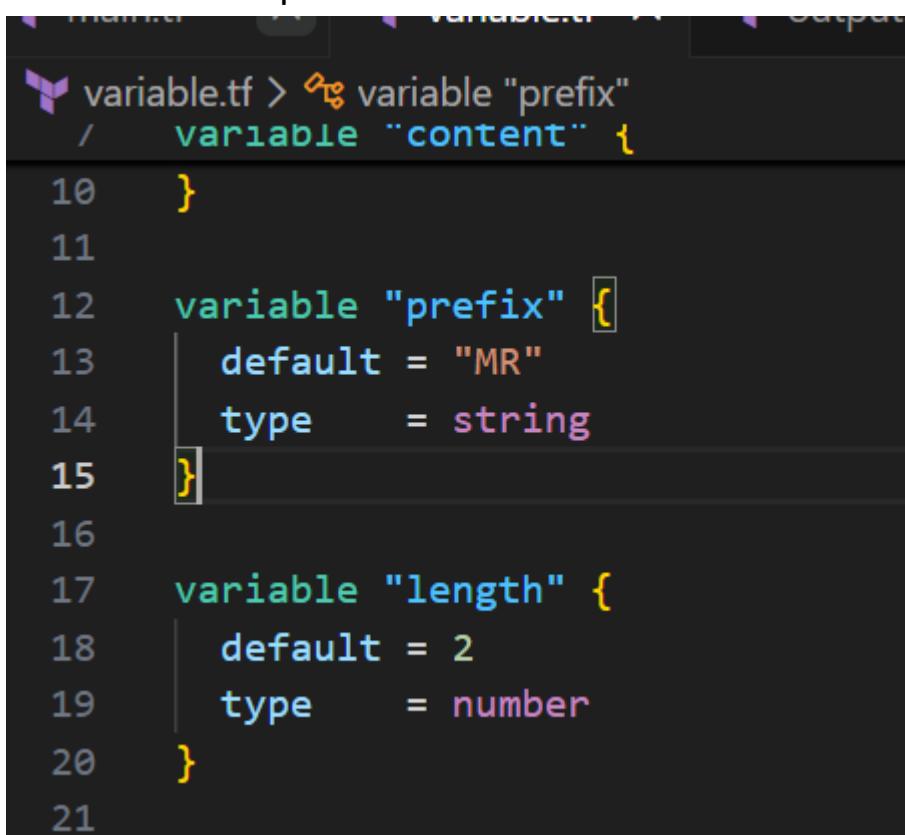
TERRAFORM 3

1. Watch the **Terraform-03** video.
2. Execute the script shown in the video.
 - Add a folder in desktop and pull it init vs code
 - Add 3 files to it main.tf
 - Variable.tf
 - Output.tf
 - Add scriptto mian.tf
 - resource "local_file" "my_pet" {
 - filename = var.filename
 - content = var.content
 - }
 -
 - resource "random_pet" "petname" {
 - prefix = var.prefix
 - separator = "-"
 - length = var.length
 - }

- 

```
main.tf > ...
1   resource "local_file" "my_pet" {
2     filename = var.filename
3     content  = var.content
4   }
5
6   resource "random_pet" "petname" {
7     prefix    = var.prefix
8     separator = "-"
9     length    = var.length
10  }
11
```

- Variable file script



```
variable.tf > variable "prefix"
/   variable "content" {
10  }
11
12  variable "prefix" {
13    default = "MR"
14    type    = string
15  }
16
17  variable "length" {
18    default = 2
19    type    = number
20  }
21
```

- Output.tf

The screenshot shows a code editor with three tabs at the top: 'main.tf', 'variable.tf', and 'output.tf'. The 'output.tf' tab is active, displaying the following Terraform code:

```
output "pet_name" {
  value = random_pet.petname.id
}
```

- Now open terminal add and give commands
- Terraform init
- Terraform plan and
- Terraform apply

The screenshot shows a terminal window with the following output:

```
random_pet.petname: Creating...
local_file.my_pet: Creating...
random_pet.petname: Creation complete after 0s [id=MR-proper-bass]
local_file.my_pet: Creation complete after 0s [id=aa9f05f39211ea80c845af77b88de87
3f63b14af]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:
```

- Now a file will be created

The screenshot shows a code editor with four tabs: 'main.tf', 'variable.tf', 'output.tf', and 'pets.txt'. The 'pets.txt' tab is active, displaying the content:

```
I love cats
```

Now -----

I want to add something to already generated file

So I need to change content in variable.tf file

Add this script

```
variable "filename" {  
    default = "pets.txt"  
    type = string  
}
```

```
variable "content" {
```

```
    default = <<EOF
```

```
I love cats
```

```
I also love dogs
```

```
EOF
```

```
    type = string
```

```
}
```

```
variable "prefix" {
```

```
    default = "MR"
```

```
    type = string
```

```
}
```

```
variable "length" {
```

```
    default = 2
```

```
    type = number
```

```
}
```

Run terminal with terraform apply

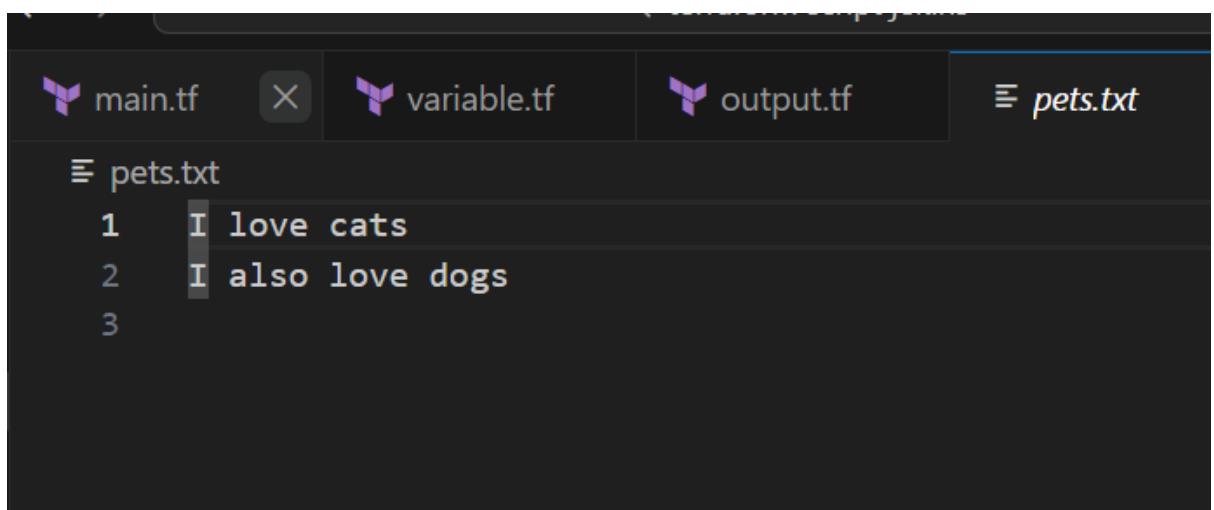
```
PROBLEMS DEBUG CONSOLE TERMINAL ... ⚙️ powershell + ⌂ ⌂ ... | ⌂ ×

random_pet.petname: Creating...
random_pet.petname: Creation complete after 0s [id=MR-dove]
local_file.my_pet: Creation complete after 0s [id=576dc285ae9aa8be49300813d87656f
f8071d23f]

Apply complete! Resources: 2 added, 0 changed, 2 destroyed.

Outputs:
```

- Output



. 3. Integrate Terraform in Jenkins using the Terraform plugin

- Login to Jenkins
 - Manage Jenkins
 - >. Pugins and install terraform

Name ↓	Health	Enabled
Terraform Plugin 1.0.10 This plugin provides a build wrapper for Terraform to launch and destroy infrastructure. Report an issue with this plugin	76	<input checked="" type="checkbox"/>

- And go to tools and add terraform

The screenshot shows the Jenkins interface under the 'Manage Jenkins' section, specifically the 'Tools' configuration. It displays four main sections: 'SonarQube Scanner installations' (with a dropdown and 'Edited' status), 'Ant installations' (with a '+ Add Ant' button), 'Maven installations' (with a dropdown and 'Edited' status), and 'Terraform installations' (with a dropdown and 'Edited' status).

4 . Create one Jenkins job using **Maven Project** for the code below with **two stages**:

- **Stage 1:** Git clone
- **Stage 2:** Maven

Compilation Code: <https://github.com/betawins/java-Working-app.git>

- Login into Jenkins
- >> pulgins install maven

s / Manage Jenkins / Plugins

The screenshot shows the Jenkins 'Manage Jenkins' > 'Plugins' page. A search bar at the top contains the text 'maven'. Below it, a table lists the 'Maven Integration plugin' version 3.27, which is described as providing deep integration between Jenkins and Maven. A note indicates it adds support for automatic trials depending on SNAPSHOTs and automated configuration of various Jenkins publishers.

- Create new item
- Select maven project in the options
- Add git clone url

Jenkins / terraform-maven-j / Configuration

Configure

Source Code Management (selected)

Git

Repositories

Repository URL: git@github.com:betawins/hiring-app.git

Credentials: git-sync

Advanced

+ Add Repository

Branches to build

Branch Specifier (blank for 'any'): */main

- And scroll to build

Build

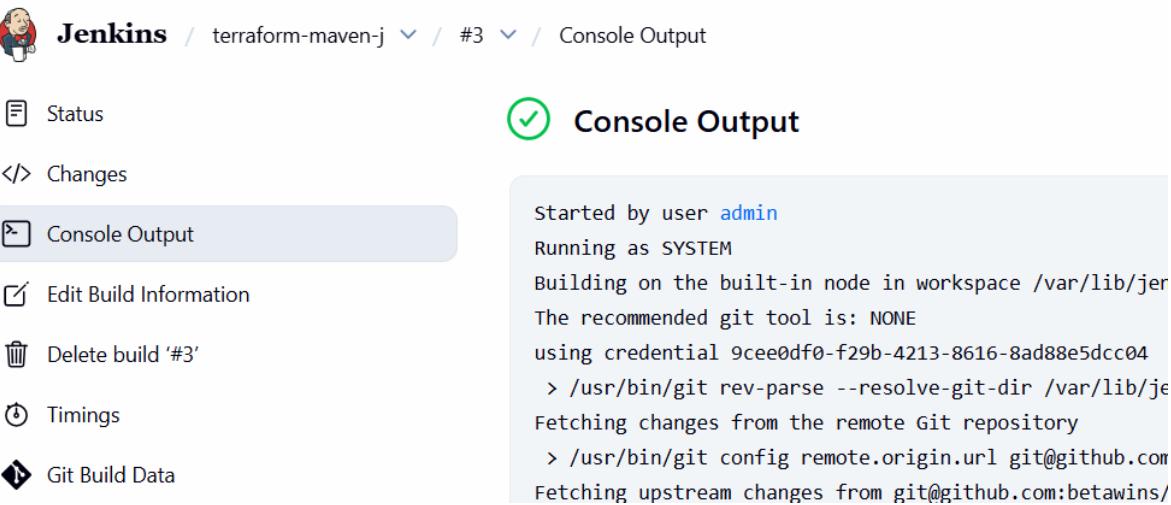
Root POM: pom.xml

Goals and options: clean compile

Advanced

- In the root coloum give POM.XML
- In th e goals give clean complie

- Run the job



The screenshot shows the Jenkins interface for a job named "terraform-maven-j". The build number is "#3". The "Console Output" tab is selected. The output log shows:

```

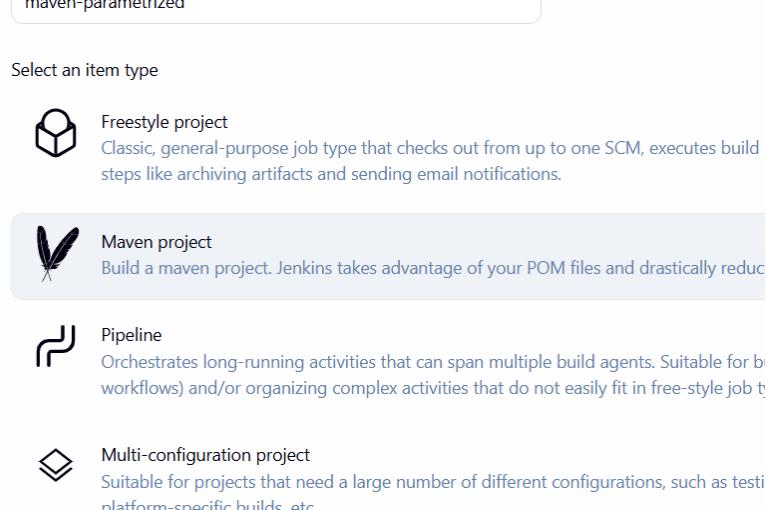
Started by user admin
Running as SYSTEM
Building on the built-in node in workspace /var/lib/jenkins
The recommended git tool is: NONE
using credential 9cee0df0-f29b-4213-8616-8ad88e5dcc04
> /usr/bin/git rev-parse --resolve-git-dir /var/lib/jenkins
Fetching changes from the remote Git repository
> /usr/bin/git config remote.origin.url git@github.com:betawins/terraform-maven-j.git
Fetching upstream changes from git@github.com:betawins/terraform-maven-j.git

```

5 . Use the same code and create a **parameterized job** in Jenkins with:

- **Stage 1:** Git clone
- **Stage 2:** Maven Compilation Code: Use the same code and create a **parameterized job** in Jenkins with:
<https://github.com/betawins/java-Working-app.git>

- Select maven project

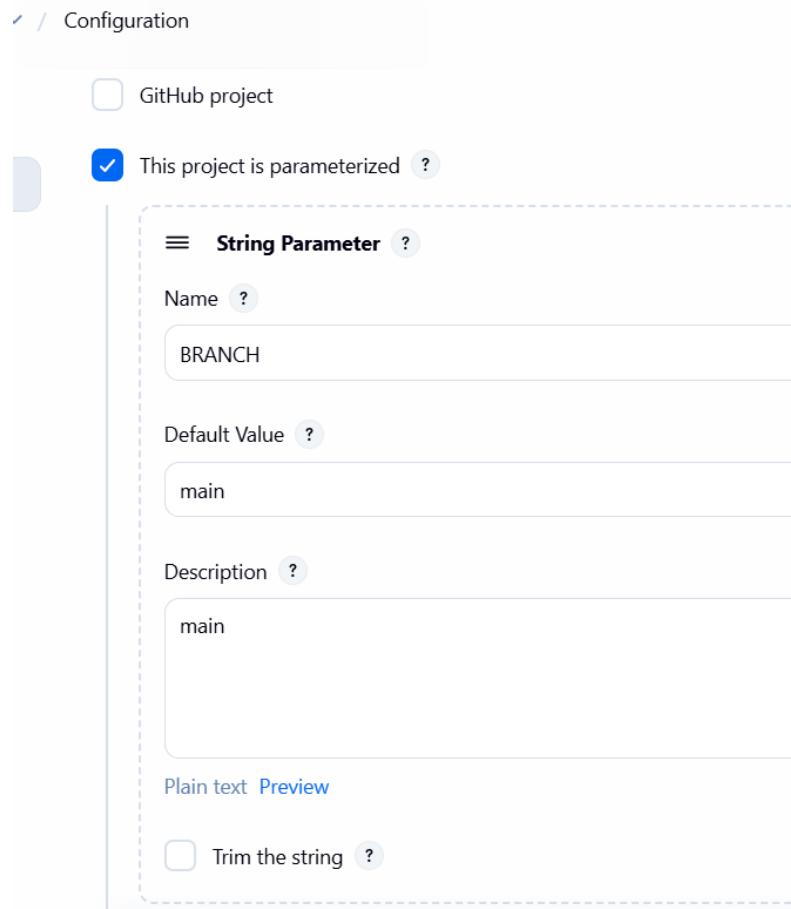


The screenshot shows the Jenkins "New Item" creation dialog. The search bar contains "maven-parametrized". The "Select an item type" dropdown is open, showing the following options:

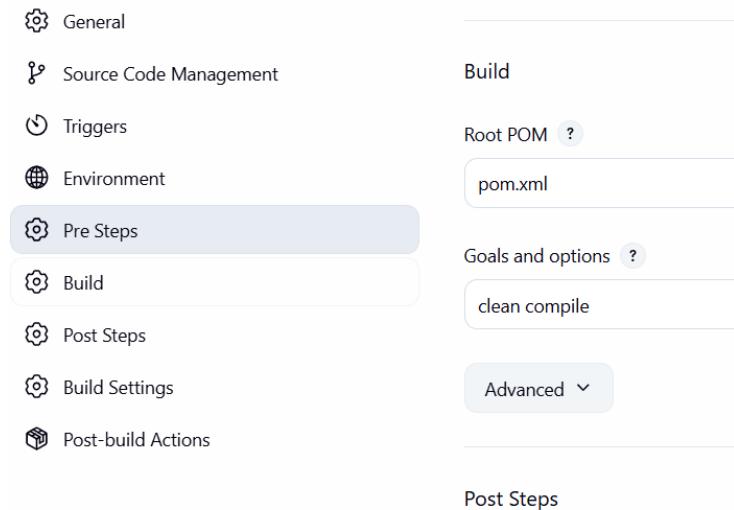
- Freestyle project**: Classic, general-purpose job type that checks out from up to one SCM, executes build steps like archiving artifacts and sending email notifications.
- Maven project**: Build a maven project. Jenkins takes advantage of your POM files and drastically reduces configuration.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building workflows and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**: Suitable for projects that need a large number of different configurations, such as testing platform-specific builds, etc.

- Now scroll to this project is parameterized.

- Select string parameter



- Add git clone url
- Now scroll to build
- And in root add POM.XML
- Now, in goals give clean comple



- Output

Jenkins / maven-parametrized #2 / Console Output

Status Changes

Console Output

Started by user admin
Running as SYSTEM
Building on the built-in node in workspace /v
The recommended git tool is: NONE
using credential 9cee0df0-f29b-4213-8616-8ad8
> /usr/bin/git rev-parse --resolve-git-dir /
Fetching changes from the remote Git repository
> /usr/bin/git config remote.origin.url git@
Fetching upstream changes from git@github.com

6 . What are the **global variables** in Jenkins?

✓ Jenkins Global Variables – Full Table

1 Built-in Environment Variables

Variable	Description
BUILD_NUMBER	Current build number (auto increments)
BUILD_ID	Unique build ID
BUILD_DISPLAY_NAME	Display name of the build
BUILD_URL	URL of this build in Jenkins

Variable	Description
JOB_NAME	Name of the Jenkins job
JOB_BASE_NAME	Job name without folder path
WORKSPACE	Full workspace directory path
NODE_NAME	Jenkins agent (node) name
NODE_LABELS	Labels assigned to the node
JENKINS_HOME	Jenkins home directory
JENKINS_URL	Base URL of Jenkins
EXECUTOR_NUMBER	Executor ID that is running the job

2 Git / SCM Variables

Variable	Description
GIT_COMMIT	Git commit hash checked out
GIT_BRANCH	Git branch name
GIT_URL	Git repository URL
GIT_PREVIOUS_COMMIT	Previous commit hash
GIT_PREVIOUS_SUCCESSFUL_COMMIT	Last successful commit

3 Pipeline Global Objects

Variable	Description
env	Access all environment variables
params	Access job parameters
scm	Auto SCM checkout configuration
currentBuild	Provides build details (status, ID, result)
docker	Use Docker containers inside pipeline
pipeline	Declarative pipeline root object
steps	Access pipeline steps
agent	Defines where pipeline runs

 **Job Parameter Variables (when using “This project is parameterized”)**

Variable	Description
<code> \${PARAM_NAME}</code>	Any String, Choice, Boolean parameter
<code> \${BRANCH}</code>	Example branch parameter
<code> \${GOALS}</code>	Example Maven goals parameter
<code> \${ENVIRONMENT}</code>	Example env parameter (dev/test/prod)

5 Shared Library Global Variables

Variable	Description
Custom global FN (e.g. hello())	Provided by Jenkins Shared Libraries under vars/
Custom classes	Reusable logic across pipelines

★ BONUS: Short “MOST IMPORTANT” Table (for interviews)

Category Important Variables

Build Info BUILD_NUMBER, BUILD_ID, BUILD_URL

Job Info JOB_NAME, WORKSPACE, NODE_NAME

Pipeline env, params, scm, currentBuild

Git/SCM GIT_COMMIT, GIT_BRANCH, GIT_URL